# ERAMCO SYSTEMS

# ES-41
# DATABASE 85

## Owner's Manual

**A Database Oriented Operating System for the Eramco Systems RAM Storage Unit**

January 1986

Printing History

Edition 1.....January 1986

Editing &
Artwork by: SoftWord,
Rectory Lane,
Windlesham,
GU20 6BW,
England.

Printed by: Dinky Druk BV.
Keizerstr. 44
1783 LP Den Helder
The Netherlands

## *** NOTICE ***

ERAMCO SYSTEMS makes no expressed or implied warranty with regard to the examples, keystroke procedures and program material offered or their merchantability or their fitness for any particular purpose. The examples, keystroke procedures and program material are made available solely on an "as is" basis, and the entire risk as to quality and performance is with the user. Should the examples, keystroke procedures or program material prove defective, the user (and neither ERAMCO SYSTEMS nor any other party) shall bear the entire cost of all necessary correction and incidental or consequential damages in connection with or arising out of the furnishing, use, or performance of the examples, keystroke procedures or program material

# CONTENTS


| Section | Contents | Page |
|---|---|---|
| i | Contents | 3 |
| ii | Acknowledgments | 4 |
| iii | Introduction | 5 |
| iv | Manual Organisation | 7 |
| 1 | Memory Structures | 15 |
| 2 | System Functions | 19 |
| 3 | Data File Functions | 33 |
| 4 | Data File Register Operations | 45 |
| 5 | Moving Data Files | 71 |
| 6 | Data File Pointer Operations | 89 |
| 7 | Data File Comparison | 99 |
| 8 | Extended Memory Functions | 119 |
| 9 | Main Memory Functions | 127 |
| 10 | Program Functions | 133 |
| 11 | Program Examples | 141 |
| | APPENDIX | |
| A | Owner's Information<br>RSU Maintenance<br>Operating Precautions<br>Warranty & Servicing | 147 |

# ACKNOWLEDGMENTS

The **ES-41 DATABASE SYSTEM** directly results from complaints from a number of HP-41 users all over the world. These users, who complained of a lack of storage space in the HP-41, users have encouraged me to develop this system, consisiting of the **ES-41 DATABASE module** and the companion **RAM Storage Unit** (RSU), as a means of data storage.

I wish to especially thank two members of the Dutch user group in Amsterdam, without whom this module and manual could not have been written. My thanks Toby Koenders and Gerlof Donga.

Also I want to sincerely thank Jan Buitenhuis, who has made this whole project possible by his continuous support and gathering of information from contacts throughout the world.

Last, but not least, I want to thank a good friend of mine, who has continued to encourage me to complete the project, even when the going became difficult. Thanks Edwin.

# INTRODUCTION

No matter whether you consider the HP-41 a calculator or a computer, the fact remains that it is currently the most powerful functionally expandable handheld computing system available.

The demands of HP-41 users have grown quite extensively since the machine's introduction in 1979. Such demands were quickly recognised by Hewlett-Packard, who responded by introducing extra plug-in-modules and advanced 41-models containing additional functions and memory. However, as users gained experience with the HP-41, so their demand for handling even larger amounts of data and application programs increased. In addition, it was soon realised that a different and enhanced operating system was needed to allow the handling of large databases.

To satisfy these demands **ERAMCO SYSTEMS**, who have been active for a number of years in the development of additional memory for the HP-41, have developed the **RAM Storage Unit** (RSU).

The **RSU** provides the 41-user with over four times the storage capacity of the Extended Functions/Memory modules. With a second **RSU**, this capacity can be increased to over 6 times.

To provide an effective means of utilising the additional memory, and allow the user to handle very large database applications, **ERAMCO SYSTEMS** have developed the **ES-41 DATABASE** operating system.

# ES-41 DATABASE SYSTEM

The **ES-41 DATABASE** System has been specifically designed for use with the HP-41 Handheld Computer and comprises the **ES-41 DATABASE Software** module and the **RSU** (RAM Storage Unit). Two models are available:

* 8K-byte RSU with inbuilt **ES-41 DATABASE** Software; or
* 16K-byte RSU with the **ES-41 DATABASE** Software contained in a separate plug-in module.

The **ES-41 DATABASE Software** contains all the functions necessary to fully exploit the capabilities of the ERAMCO SYSTEMS' RAM Storage Unit as an additional memory device. Functions included can be used to create your own HP-41 "module" of programs.

Other commands in the function set implement a complete data file system. This data file system now provides you with over 2700 data storage registers on your HP-41. To facilitate database handling of these registers, functions are provided to quickly and easily locate, manipulate, retrieve and store a specific data item in any one of the 2700 registers.

Finally, the ease of using the HP-41 is increased by new functions dealing with Extended Memory and the HP-41 as a whole.

For a good understanding of how to operate all the **ES-41 DATABASE** functions it is essential to read the manual carefully. In many cases, we have illustrated the functions usage by including example programs and register diagrams. If you should ever find difficulty in understanding the function descriptions, working through these examples and diagrams will be sufficient to provide total understanding.

We sincerely hope the **ERAMCO SYSTEMS RSU**, together with the **ES-41 DATABASE** Software, will fulfil all your database and memory requirements. Should you have suggestions for improvement, please feel free to pass them on to us.

# MANUAL ORGANISATION

This manual contains a complete description of the instruction set provided by the **ES-41 DATABASE** module. For ease of understanding, the instruction set has been subdivided into 9 classes of functionally or operationally related instructions. Each class of instruction has been described in a separate manual section. In some instances, functions have been further grouped according to operating characteristics. Where a function could logically occur in a number of different classes, it has been described only once, at the first occurrence, and referenced in other classes.

Manual sections 3 through 7 all describe functions specific to data file operations and should be carefully read as a group for a good understanding of the strengths of the **ES-41 DATABASE** operating system.

The final three sections (Sections 8 through 10) can be read individually with reference to Section 2. Functions described in these sections perform tasks that can be executed independently of functions described in other sections.

In writing this manual we have assumed a familiarity with programming the HP-41 and an understanding of the terminology covered in the various HP-41 Owner's Manuals. This manual will not tell you how to use the HP-41 itself. New HP-41 users should therefore be prepared to learn to use the HP-41 before using the **ES-41 DATABASE** module.

Program examples and keystroke procedures in this manual follow a similar format to the HP manuals and will be familiar to most users. The following hints may help you get started:

quote marks (e.g. "TESTING")

These indicate ALPHA characters. To key these in, first press the [ALPHA]-key, then key in the individual characters, and lastly press the [ALPHA]-key again.

functions (e.g. SIZE)

Not all functions are directly available on the keyboard. If not, you should use the execute function. To do so, first press the [XEQ]-key. Then press the [ALPHA]-key and spell out the function's name. Finally, press the [ALPHA] again. The function will then either execute immediately, or be entered as the next program step depending upon whether or not you are currently in PRGM-mode.

For a frequently used function, you could consider using the key assignment (ASN) facility in the HP-41 to assign the function to a USER-mode key. Refer to the HP-41 Owner's Manuals.

## Function Descriptions

Function descriptions conform to the following outline:

### FUNCTION — Brief description of the function's purpose

**XROM number**

Details: An explanation of the purpose and operation of the function.

Cautions: Warnings regarding actions the user should avoid, or measures to take to avoid difficulties.

Input: The parameters or values the user is required to supply to the Alpha, Stack or other register.

Output: Details of prompts the function displays, and the values left in registers following execution.

Example: An example program or key sequence showing:
*keysteps display comments*

Errors: Details of error messages displayed by functions. Refer to the *HP-41 Owners Manuals* for details of standard HP-41, HP-IL device messages.

To provide an overview of the **ES-41 DATABASE** instruction set, the following contains a short description of the contents and basic operations of each section of the manual.

# The ES-41 DATABASE Module Instruction Set

## System Functions section 2

**Section 2** contains three subsections, each dealing with another specific aspect of system control.

| | |
|---|---|
| **CLRSU** | CLear RSU |
| **INIDATA** | INItialize DATA block |
| **INIPRGM** | INItialize PRoGraM block |
| | |
| **DELFL** | DELete FiLe |
| **FLDIR** | FiLe DIRectory |
| **RENFL** | REName a FiLe |
| **RLEFT?** | Room LEFT ? in rsu |
| | |
| **READRAM** | READ RAM image from tape to rsu |
| **WRTRAM** | WRiTe RAM image to tape |

With the first three functions, the **RSU** can be initialized to its default state. In normal operation these functions will only need to be executed once. These functions could be compared to the NEWM (NEW Medium) function which is used for initialising mass storage media such as the HP82161A Digital Cassette.

The next four functions perform file maintenance duties.

The last two functions perform transfer duties (store and retrieve) of moving RSU-images between the **RSU** and mass storage media such as digital cassette and disc.

## Data File Functions section 3

Section 3 contains functions operating on an entire data file. In some cases care should exercised when using these functions:

| | |
|---|---|
| **CLDF** | CLear Data File |
| **CRDF** | CReate Data File |
| **RESDF** | RESize Data File |
| **RSTDF** | ReSeT Data File |
| **RSTDFX** | ReSeT Data File to X |
| **SZ?DF** | SiZe ? of Data File |

## Register Operations section 4

Section 4 contains functions working on a single register at a time within the working or current data file. There are four separate sub-groups:

| | |
|---|---|
| **DF+** | Data File-register addition |
| **DF-** | Data File-register subtraction |
| **DF*** | Data File-register multiplication |
| **DF/** | Data File-register divide |
| | |
| **RCLDF** | ReCaLl Data File register |
| **RCLDFX** | ReCaLl Data File register by X |
| **RCLIDF** | ReCaLl df-register and Increment the Data File pointer |
| | |
| **STODF** | STOre x in Data File |
| **STODFY** | STOre x in Data File at Y |
| **STOIDF** | STOre x at df-pointer and Increment the Data File pointer |
| | |
| **X<>DF** | exchange X with Data File register |
| **X<>DFY** | exchange X and Data File register at Y |

The first subgroup enables you to modify a value within a data file register without the need to recall it first, perform calculations with it and then restore it back again.

The second subgroup allows you to retrieve data.

The third subgroup handles the storage of data into a data file.

The last subgroup deals with exchanging the contents of a register.

## Moving Data Files section 5

**Section 5** is divided into three subgroups and contains functions for working with blocks of data at once.

| | |
|---|---|
| **EXREG** | EXchange REGisters |
| **RCLREG** | ReCaLl to REGisters |
| **STOREG** | STOre REGisters |
| | |
| **EXREGX** | EXchange REGisters by X |
| **RCLREGX** | ReCaLl REGisters by X |
| **STOREGX** | STOre REGisters by X |
| | |
| **EXST** | EXchange STack registers |
| **RCLST** | ReCaLl to STack registers |
| **STOST** | STOre STack registers |

The first group apply to all main memory data registers and affect a block of the same size in the RSU Data File.

The second group of functions deal with a specified number of registers in main memory and affect the same number of registers in the RSU Data File.

The last group always work on five registers at a time, namely the stack registers (X, Y, Z, T and LASTX registers).

## Pointer Operations section 6

Section 6 contains the functions that give you control over the Data File pointer.

| | |
|---|---|
| **DPTDF** | Decrement PoinTer Data File |
| **IPTDF** | Increment PoinTer Data File |
| **MPTDF** | Move PoinTer Data File |
| **RPTDF** | Recall PoinTer Data File |
| **SPTDF** | Set PoinTer Data File |
| **X=PT?** | X-register equals PoinTer ? |

## Data File Comparison section 7

Section 7 is divided into two subgroups:

| | |
|---|---|
| **FIND** | FIND target value |
| **FINDI** | FIND target value Inverse |

| | |
|---|---|
| **X=DF?** | X = Data File register ? |
| **X<>DF?** | X <> Data File register ? |
| **X<DF?** | X < Data File register ? |
| **X<=DF?** | X <= Data File register ? |
| **X>DF?** | X > Data File register ? |
| **X>=DF?** | X >= Data File register ? |

The first subgroup enables searching of a Data File for any desired numeric or string value.

The second subgroup contains six comparison functions to compare the value in the X register with a register in the Data File.

## Extended Memory Functions section 8

**Section 8** contains functions for interacting with the complete range of extended memory.

| | |
|---|---|
| **CLEM** | CLear Extended Memory |
| **EXEM** | EXchange Extended Memory |
| **RCLEM** | ReCaLl Extended Memory |
| **STOEM** | STOre Extended Memory |

## Main Memory Functions section 9

**Section 9** contains functions for interacting with the entire contents of the calculator's main memory.

| | |
|---|---|
| **EXALL** | EXchange ALL main memory |
| **RCLALL** | ReCaLl ALL main memory |
| **STOALL** | STOre ALL main memory |

These functions now make it possible to have various configurations, applications or different calculators available in your **RSU**. For example, with one function you could switch between a financial and a technical calculator. In addition, these functions can be very useful should you wish to save the calculator's memory untouched whilst you perform general calculations during the running of an application program.

## Program Functions section 10

Section 10 is divided into two subgroups and contains the functions for working with user code programs (RPN) in a RSU block.

| | |
|---|---|
| **CLPR** | CLear Program in Rsu |
| **LOADP** | LOAD Program to program block |
| | |
| **CLSEC** | CLear SECured status |
| **SETSEC** | SET prgm block SECured |

The first group facilitates the transfer of a user code program from main memory to the first empty, unsecured Program Block within the RSU. One function loads a program while the other clears a program from a RSU Program Block to allow the updating of programs within a RSU block.

The second group has two functions to control whether editing (loading and deletion) on a specified program block is possible.

Programs may be copied out of the RSU-program block, back into main memory, by means of the standard HP-41 **COPY** function.

# Section 1

## MEMORY STRUCTURES

To handle the greater memory capacity available with the RSU, the **ES-41 DATABASE** module employs a slightly different structure for memory organisation. The memory structure built with this module can be classified in two major units:

the **Block**, and the **File**.

**Block:** this is the largest unit of RSU memory organisation, and represents the same amount as a HP-41 plug-in module (equivalent to 4K-bytes) such as the GAMES Module.

Although the HP-41's four I/O (Input/Output) Ports can physically hold four plug-in modules, each port can address two blocks (also known as banks or pages) of 4K-bytes - an upper and a lower block.[1] Because the **16K-byte RSU** has more memory than can be addressed into one HP-41 I/O port, the box contains address switches so that the user can control in which ports the **RSU** memory actually appears. This is particularly useful when the user has port independent modules (the TIMER, PRINTER, HP-IL or Memory modules) plugged into the HP-41 ports. Although these modules physically consume an I/O port, they are actually addressed into the system, rather than port addressed. These spare port addresses can therefore be used by the RSU. (See the *ERAMCO SYSTEMS RAM Storage Unit Manual* for details about port addressing.)

---

1: For Advanced Users. Some of the newer plug-in modules now employ a technique of bank or page switched modules. With this electronic technique, additional banks are automatically switched in and out as required; thereby enabling the two bank (8K) limit on each port to be exceeded.

The HP-41 allows a memory structure with a theoretical maximum of 8 blocks of 4K-bytes to be created when two 16K-byte RSU are connected. However, as at least two of these blocks are needed for the 8K-byte **ES-41 DATABASE** module itself, in practice only 6 blocks may be used.

Each block in the **ES-41 DATABASE** structure is independent of other blocks and may be removed or appended at will. But, before doing so, the user should make sure that a block to be appended does not contain files or programs with the same name as others already present in the structure.

The **Block** can be further classified into:

the **Program Block**, and the **Data Block**.

**Program Block:** This is a block that has been initialised with a block header that will appear in a **CAT**alogue **2** listing in much the same way that an application module header will appear. Once initialised, the user may load and run his RPN programs from the Program Block as if they were in a custom application module. Program Blocks also possess an XROM number similar to those of a plug-in module.

**Data Block:** This is a block that has been initialised to accept one or more data type files.

Each data block in the file structure is capable of holding one file with a maximum of 679 registers, or more files where each is composed of a smaller number of registers. Adding a new file to the file structure, causes the file to be inserted into the first data block in the structure with enough free registers to hold a file of the desired length. This means that listing the data files using the function **FLDIR** (FiLe DIRectory) may not display them in order of creation. However, within any one data block, the order of display will always be the order of creation.

**File:** The file is the effective unit of data storage within the memory structure and consists of a number of registers in the same manner as a normal HP-41 file.

The structure of a file consists of its name, contained in one register, followed by one third of a register containing information about the particular type and the length of the file. The actual registers containing the information in the file begin thereafter.

Because the space occupied by a file is measured in registers, it is easy to calculate how many files can be held in a data block, or how much space is left in a certain data block.

When a data block is initialized, the number of free registers in that data block is set to 679. This is the number of registers available for data storage and indicates that it is possible to create a file of 679 registers in length. The file maintenance functions always take account of the number of registers needed to store the name and the file information. This means that in reality there are 680 and one third registers available within an initialized data block.

After creation of the first file within a data block, file header data is compacted into the registers used to store header information relating to file type and length. Therefore after the first file is created, only one more register is needed to create the next three files. After the first three files have been created another register will be used to contain the entries for the next three files.

The file and register usage is perhaps more clearly explained by means of an example.

> After initialisation of a data block, the number of free registers is 679. Creating a file called "ABC" of 10 registers length will reduce the available registers to 667 and not 669 as you might have expected. This is in accord with the description above. There are 10 registers used for the file, one register is reserved for the filename of the next file and one register is used

to hold the information about file type and length for the next three file entries.

If we again create a file of 10 registers length, we will have 656 registers left. We used 10 registers for the file and reserved one register for the next file name.

Creating a third file of 10 registers in length will again consume 11 registers. This amounts to 10 registers for the file and one register reserved for the next file name.

Creating a fourth file of 10 registers will use up the 10 registers allocated to the file, plus one register for the file name and one register for the file type and length of the next three files.

Although this may sound complicated, the number of really free registers within a data block is automatically taken care of by the file management system of the **ES-41 DATABASE** module. In practice, you will never have to concern yourself with it.

# Section 2

# SYSTEM FUNCTIONS

**System Functions** are concerned with aspects of controlling the **RAM Storage Unit (RSU)** and can be classified into three groups.

Three of the functions:

| | |
|---|---|
| **CLRSU** | CLear RSU |
| **INIDATA** | INItialize DATA block |
| **INIPRGM** | INItialize PRoGraM block |

initialise the **RSU** to its default state. In normal operation these functions will only need to be executed once, when a RSU-block is first used, or when, for some particular reason, you desire to clear the complete **RSU** (all blocks). These functions are comparable with the NEWM (NEW Medium) function which is used for initialising mass storage media.

The next four functions:

| | |
|---|---|
| **DELFL** | DELete FiLe |
| **FLDIR** | FiLe DIRectory |
| **RENFL** | REName a FiLe |
| **RLEFT?** | Room LEFT ? in rsu |

perform essential file maintenance duties.

The last two functions:

| | |
|---|---|
| **READRAM** | READ RAM image from mass storage |
| **WRTRAM** | WRiTe RAM image to mass storage |

perform file transfers, store and retrieval, between the **RSU** and mass storage media such as digital cassette or disk.

## System Function Parameters:

Most of these system functions require the user to specify the particular RSU-block on which the action is to take place. Wherever the user is asked to specify the RSU-block in the X-register, a common format is used and corresponds to the port number to which the RSU-block is addressed. According to a user's own particular set up, this may, or may not, be the port number into which the RSU is plugged. (See the *ERAMCO SYSTEMS - RAM Storage Unit Manual*).

The particular RSU-block within the port (there are two blocks within a port) is indicated by a positive or negative numeric value of between 1 and 4. A lower port block is indicated by a positive value (+1 to +4), whilst an upper port block is indicated by a negative value (-1 to -4).

| Port 1 | Port 2 |
|---|---|
| Upper Block (-1) | Upper Block (-2) |
| Lower Block (+1) | Lower Block (+2) |
| **Port 3** | **Port 4** |
| Upper Block (-3) | Upper Block (-4) |
| Lower Block (+3) | Lower Block (+4) |

# CLRSU

**CLear all RSU-boxes**

**XROM 04,04**

**Details:** Any RSU currently connected to the HP-41 will be irrevocably cleared and all data and programs destroyed. The message "CLEAR RSU" will be displayed whilst the clearing operation takes place.
To clear a particular RSU-block, either the **INIDATA** or **INIPRGM** functions should be used. Blocks initialised for one purpose may be re-initialised (and cleared) for the other purpose.

**Cautions:** **Be careful with this function as there is no recovery. It will clear all stored information from ANY connected RSU even though the port block enable switches inside the RSU may be physically set to the off position.**
Be patient during execution as this takes about 30 seconds.

**Input:** None

**Output:** The message "CLEAR RSU" will be displayed during execution.
To avoid problems, the working file will become unspecified.

**Errors:** None

# INIDATA

**INItialise DATA block**

**XROM 04,21**

Details: The RSU-block specified in the X-register will be cleared and initialised into a **data block.**
The specified port block must be between -1 to -4, or +1 to +4. Where a positive value indicates a lower port block and a positive value indicates an upper port block.
Following execution, the working file becomes undefined.

Cautions: **This function will destroy all data and programs currently stored in the specified block.**

Input: The X-register should contain the numeric value between -1 to -4, or +1 to +4 to indicate the RSU port block to be initialised.

Output: The working file becomes undefined.

Errors: Refer to the *HP-41 Owners Manuals* for details of displayed standard messages.

"NO RSU PAGE" A RSU is either not connected at the specified port block, or the specified port block has been disabled.
The specified port number may contain a plug-in RAM or ROM module rather than a RSU-block.

"BAD PAGENO" The X-register contains an illegal block number. Limits are: -1 to -4 and +1 to +4.

"NONEXISTENT" An illegal number, less than -999 or more than +999, was specified in the X-register.

# INIPRGM

**INItialise a PRoGraM block**

**XROM 04,22**

Details: The specified RSU-block will be cleared and initialised as a **program storage block.**
The name specified in the ALPHA-register will become the CATalogue 2 header for the program block. All subsequent programs loaded into the program block will appear in CAT 2 following this header.
The XROM (External ROM) number assigned to the block will be used as the function reference for all programs and global labels within all programs in that program block. *See Section 10.*

Cautions: **This function will destroy all data or programs currently stored in the specified block.**
**The user should ensure that the assigned XROM number does not conflict with other plug-in modules present in the HP-41, or that the user may use in the future.**

Input: The Alpha-register should contain the alphanumeric name (max. 11 characters) for the program block.
The X-register should contain the RSU-block number. The user may specify lower port blocks by positive values (+1 to +4), or may specify upper port blocks by negative values (-1 to -4).
At the prompt message:

INIPRGM _ _ _

the user should key in an XROM number to be assigned to the program block (a value between 001 and 031).

Output: The working file becomes undefined.

Errors: Refer to the *HP-41 Owners Manuals* for details of displayed standard messages.

"NO RSU PAGE" RSU is either not connected to the specified port block, or the RSU-block at this page is currently disabled. The specified port number may contain a plug-in RAM or ROM module rather than a RSU-block.

"BAD PAGENO" The X-register contains an illegal block number. Limits are: -1 to -4 and +1 to +4.

"BAD XROMNO" An illegal XROM value was keyed in at the prompt. Limits are: 1 to 31. The specified XROM value is the same as that of another plug-in module, or RSU-program block currently plugged into the HP-41.

"NAME ERROR" The Alpha-register does not contain a name for the program block, or the specified name exceeds 11 characters in length.

"NONEXISTENT" An illegal number, less than -999 or more than +999, was specified in the X-register.

# DELFL

**DELete FiLe**

**XROM 04,07**

**Details:** Deletes the specified file from the RSU. The RSU data block containing the specified file will be packed to free the maximum number of registers for further files.

**Input:** The Alpha-register should contain the name of the data file to be deleted. Names may consist of 1 to 7 alphanumeric characters.

**Output:** After execution, the working file becomes undefined.

**Errors:** Refer to the *HP-41 Owners Manuals* for details of displayed standard messages.

"NOT A FILE" The name of an existing file was not specified in the Alpha-register. The RSU-block containing the file may not currently be enabled.

"NAME ERROR" An illegal file name was specified in the Alpha-register. Legal file names consists of between one and seven characters.

"NO FILE NAME" No file name was specified in the Alpha-register

# FLDIR

**FiLe Directory**

**XROM 04,20**

Details: Displays a directory of data files in any RSU currently connected to the HP-41.
Directory listings show:

*filetype, number of registers, filename.*

The last file name displayed will be left in the Alpha-register.
Pressing and holding down any key, except the [ON] key, will freeze the currently displayed file entry in the display until the key is released.
The directory terminates with a value in the X-register showing the highest number of free data file registers left in any data block.
If an enabled printer is attached to the HP-41, a printed directory will be obtained. Pressing the [R/S] key terminates the listing.

Input: None

Output: Following the directory display, the name of the last file entry will be left in the Alpha-register and the X-register will show the number of free registers left in the RSU-block containing the largest number.
Data files may not be listed in order of creation because the file management system utilises free registers wherever possible.

Errors: Refer to the *HP-41 Owners Manuals* for details of displayed standard messages.

"NO RSU PAGE" No RSU-block is currently enabled. The RSU-system contains only program pages, but no data block pages. Create a data block using the INIDATA function.

# RENFL

**REName a FiLe**

**XROM 04,35**

**Details:** Renames the specified file from the "*old name*" to the "*new name*".

**Input:** The Alpha-register should contain a text string of the format:

*OLDNAME , NEWNAME*

where the "," is used as a separator.
Both the *old* and *new* file names must be between 1 and 7 characters in length.

**Output:** None

**Errors:** Refer to the *HP-41 Owners Manuals* for details of displayed standard messages.

"NO FL NAME" A file name was not specified in the Alpha-register.
The specified data file name "*oldname*" does not currently exist in the RSU.

"NAME ERROR" The specified *old* or *new* filename exceeds seven characters in length.
A separator "," was not placed between the *old* and *new* names.

"DUP FL NAME" The specified file name for *newname* already exists in the data storage blocks.

## RLEFT?

**Room LEFT in RSU-box**

XROM 04,37

Details: Returns to the X-register the number of free registers found in the data storage block containing the most free registers in the RSU-system.

Cautions: **The returned value will not indicate the maximum or total number of registers available within the whole RSU-system.**

Input: None

Output: The number of free registers in the data storage block with the most free registers available is returned to the X-register.
If the stack lift is enabled, the stack will be raised and the value in the T-register will be lost.
If the stack left is disabled, the previous value in the X-register is overwritten.

Example: If two data storage blocks have been initialised, one empty and the other containing data files, then **RLEFT?** will return the value "679" to the X-register. This is the number of free registers in the empty data block, not the total available in both blocks.

Errors: Refer to the *HP-41 Owners Manuals* for details of displayed standard messages.

"NO RSU PAGE" No RSU is currently connected.
The block is possibly disabled, or a data storage block has not yet been initialised.

## READRAM

**READ a RAM-file from mass storage to RSU-block**

**XROM 04,34**

Details: Reads back the contents of an entire RSU-block that was written to a mass storage medium (such as tape cassette) using the **WRTRAM** function. The tape image is automatically validated and automatically purged from the RSU if incorrect.

Cautions: **Although it is possible to read back images written to tape by functions from other modules, these images do not contain the necessary checksum and therefore will be invalid.** The **ES-41 DATABASE** function **WRTRAM** should be used to write an RSU-image to tape.

Input: The Alpha-register should contain the file name to be loaded from mass storage medium.
The X-register should contain the RSU-block number that the image is to be loaded into. The user may specify a lower port block by a positive value (+1 to +4) or an upper port block by a negative value (-1 to -4)

Output: During execution the message READRAM will be displayed.
Following execution the working file will become undefined.
If the image read from mass medium does not contain a valid checksum, the image is considered to be corrupted or misrecorded. Because corrupted data will disturb correct operation of the RSU, the image is automatically purged from the RSU-block, and the error message "BAD ROMFILE" will be displayed. The original content of the RSU-block will have been lost and is irrecoverable.
The working file becomes undefined.

Errors: Refer to the *HP-41 Owners Manuals* for details of displayed standard messages.

In addition, refer to the Owner's Manual of the appropriate device for details of HP-IL module or mass storage device error messages.

"NO HP-IL" The HP82160 HP-IL Module is not connected to the HP-41.

"BAD ROMFILE" A corrupted image was loaded from mass medium, the checksum failed to verify, and the image was purged from the RSU. Check the mass storage device and medium, then repeat the procedure. If the same error occurs, then the image is probably lost and irrecoverable.

"NO RSU PAGE" READRAM has been executed without leaving a RSU-page block available or enabled.
The specified port block number may contain a plug-in application module (ROM), not an RSU-block. Check the RSU addressing switches.

"BAD PAGENO" The port block number specified in the X-register was invalid and does not correspond to a connected and enabled RSU-block.

"NONEXISTENT" The number specified in the X-register is less than -999 or greater than +999.

# WRTRAM

**WRiTe RAM image to mass storage**

**XROM 04,52**

Details: Transfers the entire contents of a RSU-block onto mass storage medium.
The system identifier for media files created by **WRTRAM** will be **$07**.
Mass storage **DIR**ectory entries will be in the format:

*filename*__ __??,$__ __ __ __ __ __*640*

where '__' indicates a space or blank.
A unique file type has been chosen for RSU-blocks in order to ensure that other standard 41-functions cannot corrupt data.
Because of the size of the data being transferred to the mass medium, the operation may take a few minutes.

Cautions: **Ensure that the RSU-block number specified in register X is both enabled and contains valid data. The RSU-block must contain at least one data file or be initialised as a program storage block.**

Input: The Alpha-register should contain the name under which the image will be written to the mass storage medium.
The X-register should contain the number of the RSU-block that is to be written to mass storage. The user may specify a lower port block by a positive value (+1 to +4), or an upper port block by a negative value (-1 to -4).

Output: The message "WRTRAM" will remain in the display during execution. Because of the size of data being transferred, this may take a few minutes.

Errors: Refer to the *HP-41 Owners Manuals* for details of displayed standard messages.

In addition, refer to the Owner's Manual of the appropriate device for details of HP-IL module or mass storage device error messages.

"NO HP-IL" The HP82160 HP-IL Module is not connected to the HP-41. Connect the HP-IL Module, switch on the HP-41 again and re-execute the **WRTRAM** function.

"BAD PAGENO" The port block number specified in the X-register was invalid and does not correspond to a connected and enabled RSU-block. Change the X-register value to between -4 and +4 excluding 0.

"NONEXISTENT" The number specified in the X-register is less than -999 or greater than +999.

# Section 3

# DATA FILE FUNCTIONS

Section 3 describes functions operating on data files in their entirety.

| | |
|---|---|
| **CLDF** | CLear Data File |
| **CRDF** | CReate Data File |
| **RESDF** | RESize Data File |
| **RSTDF** | ReSeT Data File |
| **RSTDFX** | ReSeT Data File to X |
| **SZ?DF** | SiZe ? of Data File |

# CLDF

**CLear Data File**

**XROM 04,01**

Details: The data file specified in the Alpha-register will be cleared of all data and all the RSU-block registers allocated to the named file will be set to zero. Because the file is cleared, and not deleted, the currently defined working data file remains unchanged.

Input: The Alpha-register should contain the name of the data file that is to be cleared.

Output: The data file is cleared and each register's content replaced by zero.
The defined working file remains unchanged.

*Before* *After*

Example:

| Prgm Step | comments |
|---|---|
| LBL "CLEARF" | |
| AON | Activates Alpha-mode |
| FILE? | |
| PROMPT | Asks for file name input |
| AOFF | Cancels Alpha-mode |
| CLDF | Clears the RSU-data file |
| RTN | |

Errors: Refer to the *HP-41 Owners Manuals* for details of displayed standard messages.

| | |
|---|---|
| "FL TYP ERR" | The file specified in Alpha is not a data type file. Use the FLDIR function to check that the file exists. |
| "NAME ERROR" | An illegal file name was specified in the Alpha-register. File names are between 1 and 7 characters in length. |
| "NO FL NAME" | A file name was not specified in the Alpha-register. |
| "NOT A FILE" | The name specified in the Alpha-register is not a valid existing file name. Use FLDIR to check all files existing. |

# CRDF

**CReate a Data File**

XROM 04,06

Details: Creates a data storage file of the name specified in the Alpha-register and of the length (number of registers) specified in the X-register.
The created data file will be placed in the first data-block with enough free register space available.
The created data file will have all allocated registers set to zero and the data file pointer set to the first element in the file (i.e. register 0).

Cautions: **A unique file name should be specified.**

Input: Alpha should contain a name for the file to be created of between 1 and 7 characters in length.
Register-X should contain the length of the data file as a number of registers. The maximum file length that can be created in an empty data block is 679 registers.

Output: A data file of the specified name and length is created, and becomes the working file.
If the function causes the "NO ROOM" error, because not enough free registers are available in any one data block, then the X-register contents will be overwritten by the maximum number of free registers available.

Errors: Refer to the *HP-41 Owners Manuals* for details of displayed standard messages.

"NO RSU PAGE" No RSU-block is currently enabled. Check the position of RSU-switches.
RSU-blocks are either not initialised, or have been initialised as program blocks, but not as data storage blocks.

"DUP FL NAME" The specified data file name currently exists. Use FLDIR to check existing names and select a new unique name.

| | |
|---|---|
| "NAME ERROR" | An illegal file name was specified in the Alpha-register. File names of 1 to 7 characters are allowed. |
| "NO ROOM" | The length specified for the file is too large for the number of free registers remaining in any one RSU-data block. Following this error message, the X-register will contain the maximum number of free registers that are available in any one data block. Either use this value as the new file length, or free additional data registers by deleting a file, or initialising a new data block. |
| "NONEXISTENT" | An illegal number, less than -999 or more than +999, was specified in the X-register. |
| "DATA ERROR" | A value of '0' registers was specified as the length of the data file in the X-register. |

# RESDF

**RESize Data File**

## XROM 04,36

Details: The data file specified in the Alpha-register is resized to the number of registers indicated in the X-register.
Where the data file is resized to a greater number of registers, all following files are moved to accommodate the new empty registers following the last register of the specified file.
Where the data file is resized to a smaller number of registers, then registers will be progressively removed from the end of the specified file towards the beginning.

Cautions: **When resizing to a smaller number of registers, the user should take care not to delete registers with data stored in them.**

Input: The Alpha-register should contain the name of the data file to be resized.
The X-register should contain the new size of the data file in registers. Note that this value is the new size, not the number of registers to remove.

Output: The working file becomes undefined.

Example: (assumes that only one data block has been initialised)

| Prgm Step | display | comments |
|---|---|---|
| "DFTEST" | DFTEST | |
| 75 | 75 | |
| RESDF | | The file in Alpha (DFTEST) is resized to 75-registers. |
| RLEFT? | 602 | 602-registers are left. |
| 150 | 150 | |
| RESDF | | File DFTEST is resized to 150-registers. |
| RLEFT? | 527 | Only 527-registers are now left. |

Errors: Refer to the *HP-41 Owners Manuals* for details of displayed standard messages.

| | |
|---|---|
| "NOT A FILE" | The data file name specified in the Alpha-register does not exist. |
| "FL TYP ERR" | The file specified in Alpha is not a data type file. Use the FLDIR function to check that the file exists. |
| "NAME ERROR" | An illegal file name was specified in the Alpha-register. File names of 1 to 7 characters are allowed. |
| "NO FL NAME" | No file name was specified in the Alpha-register. |
| "NONEXISTENT" | An illegal number, less than -999 or more than +999, was specified in the X-register. |
| "NO ROOM" | The specified length for the file is too long for the number of free registers remaining in the RSU-data block. Either free additional data registers by deleting a file (using **DELFL**), or resize another file(s). If only one data block is available, then **RLEFT?** can be used to check the number of free registers remaining. |

# RSTDF — ReSeT Data File

## XROM 04,39

Details: Reselects the specified file as the working file and sets the data pointer to the first register in that file.

Input: The Alpha-register should contain the name of the data file to be selected as the working file. File names are limited to 1 to 7 characters.

Output: None.

Example:

| Prgm Step | display | comments |
|---|---|---|
| "DFTEST" | DFTEST | Put file name (DFTEST) into Alpha-register. |
| RSTDF | | selects the file named "DFTEST" as the working file. |

Errors: Refer to the *HP-41 Owners Manuals* for details of displayed standard messages.

"NOT A FILE" The name given in the Alpha-register is not that of a currently existing file.

"FL TYP ERR" The file specified in Alpha is not a data type file. Use the FLDIR function to check that the file exists

"NAME ERROR" An illegal file name was specified in the Alpha-register. File names of 1 to 7 characters are allowed.

"NO FL NAME" No file name was specified in the Alpha-register.

# RSTDFX

**ReSeT Data File to X**

XROM 04,40

Details: Resets the data file specified in the Alpha-register to the working file and also resets the data file pointer to the value specified in the X-register. See also the **RSTDF** function.

Input: The Alpha-register should contain the name of file to become the working file.
Register-X should contain the value that the data file pointer is to be reset to.

Output: The working data file is reset to the file specified in Alpha and the pointer to the value specified in the X-register

Errors: Refer to the *HP-41 Owners Manuals* for details of displayed standard messages.

"NOT A FILE" The name given in the Alpha-register is not that of a currently existing file.

| | |
|---|---|
| "END OF FL" | The value in the X-register is greater than the size of the specified data file. Change the pointer value to a value less than the size of the data file. |
| "NONEXISTENT" | An illegal number, less than -999 or more than +999, was specified in the X-register. |
| "FL TYP ERR" | The file specified in Alpha is not a data type file. Use the FLDIR function to check that the file exists. |
| "NAME ERROR" | An illegal file name was specified in the Alpha-register. File names of 1 to 7 characters are allowed. |
| "NO FL NAME" | No file name was specified in the Alpha-register. |

# SZ?DF

**SiZe ? of Data File**

**XROM 04,51**

**Details:** Recalls the size of the current data file to the X-register.

**Input:** None

**Output:** When the stack lift is enabled, the stack will be raised and the size of the data file will be placed in the X-register. The previous value in the T-register will be pushed out of the stack and will be lost.
When the stack lift is disabled, the previous contents of the X-register will be overwritten by the size of the data file.

**Example:**

| Prgm Step | display | comments |
|---|---|---|
| CLX | 0.0000 | Clears the contents of the X-register and disables the stack lift. |
| SZ?DF | 100.0000 | Returns the number of registers in the current data file. |

**Errors:** Refer to the *HP-41 Owners Manuals* for details of displayed standard messages.

"NO WORKFL" There is no current or working file selected. Use **RSTDF** to reselect the working file.

# Section 4

# DATA FILE REGISTER OPERATIONS

**Section 4** describes functions working on single registers within the working or current data file. Register operations are divided into four separate functionally related sub-groups.

The first subgroup enables modification of a data file register value without recalling it to the stack or main memory first:

| | |
|---|---|
| **DF+** | Data File-register addition |
| **DF-** | Data File-register subtraction |
| **DF*** | Data File-register multiplication |
| **DF/** | Data File-register divide |

The second subgroup allows you to retrieve data from data file registers:

| | |
|---|---|
| **RCLDF** | ReCaLl Data File register |
| **RCLDFX** | ReCaLl Data File register by X |
| **RCLIDF** | ReCaLl df-register and Increment the Data File pointer |

The third subgroup handles storage of data into a data file:

| | |
|---|---|
| **STODF** | STOre x in Data File |
| **STODFY** | STOre x in Data File at Y |
| **STOIDF** | STOre x at df-pointer and Increment the Data File pointer |

The last subgroup deals with exchanging the contents of a register.

| | |
|---|---|
| **X<>DF** | exchange X with Data File register |
| **X<>DFY** | exchange X and Data File register at Y |

## DF+ Data File register addition

**XROM 04,08**

Details: The value in the X-register is added to the current content of the data file register
The particular data file register is specified by the current position of the data file pointer.

Input: The X-register should contain the value to be added to the data file register.
The data file pointer should be set to the register, within the current file, on which the arithmetical operation is to occur.

Output: The data file register contains the result of the addition of the X- and data file register values.
The data file pointer is not moved by the operation.

| Example: | Prgm Step | Display | Comments |
|---|---|---|---|
| | 10 | 10_ | Value to set pointer to. |
| | SPTDF | 10 | Sets pointer to DF:10. |
| | 30 | 30_ | Value to add to DF:10. |
| | STODF | 30 | Stores 30.0000 into DF:10. |
| | 15.56 | 15.56_ | |
| | DF+ | 15.56 | Adds 15.56 to contents of DF:10. |
| | RCLDF | 45.56 | Recalls new contents of DF:10. |

Errors: Refer to the *HP-41 Owners Manuals* for details of displayed standard messages.

"OUT OF RANGE" Performing the operation would cause an arithmetical over- or underflow error as the result is outside the range of the HP-41.

"ALPHA DATA" Either the data file register or X-register contains alphanumeric data and therefore cannot accept arithmetical operations.

"NO WORKFL" There is no working data file selected. Use **RSTDF** to select a working file.

"END OF FL" The current data file pointer value points past the end of the working data file. Change the pointer to a value less than the data file size, or use **RESDF** to increase the file size.

## DF- Data File register subtraction

**XROM 04,09**

Details: The value in the X-register is subtracted from the contents of the data file register.
The particular data file register is specified by the current position of the data file pointer.

Input: The X-register should contain the value to be subtracted from the data file register.
The data file pointer should be set to the register, within the current file, on which the arithmetical operation is to occur.

Output: The data file register contains the result of the subtraction of the X-register value from the data file register value.
The data file pointer is not moved by the operation.

| Example: | Prgm Step | Display | Comments |
|---|---|---|---|
| | 235 | 235_ | Value for DF pointer. |
| | SPTDF | 235.0000 | Resets pointer. |
| | 189 | 189_ | Value to be stored in DF. |
| | STODF | 189.0000 | Value stored in DF:235. |
| | 5 | 5_ | Value to be subtracted. |
| | DF- | 5.0000 | Value subtracted. |
| | RCLDF | 184.0000 | Result recalled from DF:235. |

Errors: Refer to the *HP-41 Owners Manuals* for details of displayed standard messages.

"NO WORKFL" There is no current or working data file selected. Use **RSTDF** to select a working file.

"END OF FL" The data file pointer contains a register value greater than the size of the selected file and is therefore pointing to a non-existent register. Change the pointer to a value less than the size of the data file, or increase the file size with the **RESDF** function.

"OUT OF RANGE" Performing the operation would cause an arithmetical over- or underflow error by returning a result outside the range of the HP-41.

"ALPHA DATA" Either the data file register or X-register contains alphanumeric data and therefore cannot accept arithmetical operations.

## DF* Data File register multiplication

**XROM 04,10**

Details: Multiplies the content of the data file register by the value in the X-register.
The particular data file register is indicated by the current position of the data file pointer.

Input: The X-register should contain the value by which the content of the data file register is to be multiplied.
The data file pointer should be set to the register, within the current file, on which the arithmetical operation is to occur.

Output: The data file register contains the result of the multiplication of the X- and data file register contents.
The data file pointer is not moved by the operation.

| Example: | Prgm Step | Display | Comments |
|---|---|---|---|
| | 235 | 235_ | Value for DF pointer. |
| | SPTDF | 235.000 | Resets pointer to DF:235. |
| | 5 | 5_ | Value to multiply DF:235 by. |
| | DF* | 5.0000 | DF:235 content multiplied by 5. |
| | RCLDF | 920.000 | Result recalled from DF:235. |

Errors: Refer to the *HP-41 Owners Manuals* for details of displayed standard messages.

"NO WORKFL" No working data file is selected. Use **RSTDF** to select a working file.

"END OF FL" The data file pointer contains a value greater than the size of the selected file and is therefore pointing to a non-existent register. Change the pointer to a value less than the size of the data file, or increase the file size with the **RESDF** function.

"OUT OF RANGE" Performing the operation would cause an arithmetical over- or underflow error by returning a result outside the range of the HP-41.

"ALPHA DATA" Either the data file register or X-register contains alphanumeric data and therefore cannot accept arithmetical operations.

## DF/ Data File register division

**XROM 04,11**

Details: Divides the content of the data file register by the value in the X-register.
The particular data file register is specified by the current position of the data file pointer.

Input: The X-register should contain the value by which the content of the data file register is to be divided.
The pointer should be set to the data file register to be divided.

Output: The data file register contains the result of the division by the X-register content.
The data file pointer is not moved by the operation.

| Example: | Prgm Steps | Display | Comments |
|---|---|---|---|
| | 235 | 235_ | Value for DF pointer. |
| | SPTDF | 235.0000 | Resets pointer to DF:235. |
| | 5 | 5_ | Value by which to divide DF:235 |
| | DF/ | 5.0000 | DF:235 content divided by 5. |
| | RCLDF | 184.000 | Result recalled from DF:235. |

**Errors:** Refer to the *HP-41 Owners Manuals* for details of displayed standard messages.

"NO WORKFL" There is no current or working data file selected. Use **RSTDF** to select a working file.

"END OF FL" The data file pointer contains a register value greater than the size of the selected file and is therefore pointing to a non-existent register. Change the pointer to a value less than the size of the data file, or increase the file size with the **RESDF** function.

"OUT OF RANGE" Performing the operation would cause an arithmetical over- or underflow error by returning a result outside the range of the HP-41.

"ALPHA DATA" Either the data file register or X-register contains alphanumeric data and therefore cannot accept arithmetical operations.

"DATA ERROR" An attempt was made to divide the content of the data file register by zero.

## RCLDF

**ReCaLl content of**

XROM 04,27

**Data File register**

Details: The content of the data file register is returned to the X-register.
The particular data file register is specified by the current position of the pointer.

Input: The data file pointer should be set to the register to be recalled to the X-register.

Output: The X-register contains a copy of the data file register content.
The pointer is not moved by the operation.

| Example: | Prgm Step | Display | Comments |
|---|---|---|---|
| | "TESTDF" | | File name. |
| | RSTDF | | Resets the working file. |
| | 5 | 5__ | Value to which the pointer is to be set. |
| | SPTDF | 5.0000 | Sets the data file pointer to DF:5 |
| | RCLDF | "content" | Recalls content of DF:5. |
| | 13 | 13__ | Value to move pointer by. |
| | MPTDF | 13.0000 | Increments pointer by 13. |
| | RCLDF | "content" | Recalls content of DF:18. |

Errors: Refer to the *HP-41 Owners Manuals* for details of displayed standard messages.

"NO WORKFL" There is no current or working data file selected. Use **RSTDF** to select a working file.

"END OF FL" The data file pointer contains a value greater than the size of the selected file and is therefore pointing to a non-existent register. Change the pointer to a value less than the size of the data file, or increase the file size with the **RESDF** function.

"ALPHA DATA" Either the data file register or X-register contains alphanumeric data and therefore cannot accept arithmetical operations.

"NONEXISTENT" An illegal number, less than -999 or more than +999, was specified in the X-register.

## RCLDFX — ReCaLl content of Data File register by X

XROM 04,28

Details: The data file pointer is set to the value specified in the X-register and the content of the data file register, indicated by the pointer, is copied to the X-register.

Input: The X-register should contain the value that the pointer is to be set to.

Output: The X-register will contain a copy of the data file register content.
The pointer is not moved by the operation.

| Example: | Prgm Step | Display | Comments |
|---|---|---|---|
| | "TESTDF" | | File name to be declared the working file. |
| | RSTDF | | Resets the working file |
| | 5 | 5_ | Value to set pointer to. |
| | RCLDFX | "content" | Sets the pointer to DF:5 and recalls content to X-register. |
| | RPTDF | 5.0000 | Recalls the pointer value to X-register. |
| | MPTDF | 5.0000 | Increments pointer by 5 to point to DF:10. |
| | RCLDFX | "content" | Recalls content of DF:10. |

Errors: Refer to the *HP-41 Owners Manuals* for details of displayed standard messages.

"NO WORKFL" There is no current or working data file selected. Use **RSTDF** to select a working file.

"END OF FL" The X-register contains a value greater than the size of the selected file and therefore would set the pointer to a non-existent register. Change the X-register to a value less than the size of the data file, or increase the file size with the **RESDF** function.

"ALPHA DATA" The X-register contains alphanumeric data and therefore cannot accept arithmetical operations.

"NONEXISTENT" An illegal number, less than -999 or more than +999, was specified in the X-register.

## RCLIDF

**ReCaLl content of Data File register & Increment pointer**

**XROM 04,30**

Details: The content of the data file register is recalled to the X-register and the pointer incremented by one.
The data file register to recall is specified by the current pointer position.

Input: The pointer should be set to the data file register to recall.

Output: The X-register will contain a copy of the data file register content.
If the stack lift was enabled, the stack will be raised and the original T-register value lost.
If the stack lift was disabled, the X-register will be overwritten by the content of the data file register.

| Example: | Prgm Step | Display | Comments |
|---|---|---|---|
| | 5 | 5_ | Register to which the pointer is to be set. |
| | "TESTDF" | 5.0000 | The file to be declared the working file. |
| | RSTDFX | 5.0000 | Resets the working file and sets the pointer to DF:5. |
| | RCLIDF | "content" | Recalls content of DF:5 and increments the pointer by one. |
| | RPTDF | 6.0000 | Recalls new pointer value. |
| | RCLIDF | "content" | Recalls content of DF:6 and increments the pointer by one. |
| | RPTDF | 7.0000 | Recalls new pointer value. |

Errors: Refer to the *HP-41 Owners Manuals* for details of displayed standard messages.

"NO WORKFL" There is no current or working data file selected. Use **RSTDF** to select a working file.

"END OF FL" The data file pointer contains a value greater than the size of the selected file and is therefore pointing to a non-existent register. Change the pointer to a value less than the size of the data file, or increase the file size with the **RESDF** function.

## STODF

**STOre in Data File**

**XROM 04,44**

**register**

Details: The content of the X-register will be stored at the current pointer location in the working data file.

Input: The X-register should contain the value to be stored in the data file register.
The data file pointer should specify the register to be overwritten with the X-register value.

Output: The data file contains a copy of the X-register value.
The data file pointer is not moved by the operation.

| Example: | Prgm Step | Display | comments |
|---|---|---|---|
| | CLX | 20_ | Value to set pointer to. |
| | "DFTEST" | DFTEST | Name of the data file. |
| | RSTDFX | 20.0000 | Reset "DFTEST" as the working file and set pointer to DF:20. |
| | 15 | 15_ | Value to be stored. |
| | STODF | 15.0000 | Value stored in DF:20. |
| | RPTDF | 20.0000 | Value of pointer still unchanged. |

Errors: Refer to the *HP-41 Owners Manuals* for details of displayed standard messages.

"NO WORKFL" There is no current or working data file selected. Use **RSTDF** to select a working file.

"END OF FL" The data file pointer contains a value greater than the size of the selected file and is therefore pointing to a non-existent register. Change the pointer to a value less than the size of the data file, or increase the file size with the **RESDF** function.

# STODFY

**STOre X in Data File**

**XROM 04,45**

**register at Y**

Details: The data file pointer is set to the position given in the Y-register and the content of the X-register stored at that location in the working data file.

Input: The X-register should contain the value to be stored in the data file register.
The Y-register should contain the value to which the data file pointer is to be set.

Output: The X-register value is copied to the data file register.
The data file pointer is set to the value given in the Y-register.

| Example: | Prgm Step | Display | comments |
|---|---|---|---|
| | "DFTEST" | DFTEST | Name of the data file. |
| | RSTDF | | Resets "DFTEST" as the working file. |
| | 15 | 15_ | Value to set the pointer to. |
| | ENTER↑ | 15.0000 | |
| | 5 | 5_ | Value to be stored. |
| | STODF | 5.0000 | Value now stored in DF:15. |
| | RPTDF | 15.0000 | Value of pointer still unchanged. |
| | RCLDF | 5.0000 | Recalls value stored in DF:15 |

Errors: Refer to the *HP-41 Owners Manuals* for details of displayed standard messages.

"NO WORKFL" There is no current or working data file selected. Use **RSTDF** to select a working file.

"END OF FL" The data file pointer value in register Y is greater than the size of the selected file and is therefore pointing to a non-existent register. Change the pointer to a value less than the size of the data file, or increase the file size with the **RESDF** function.

"NONEXISTENT" An illegal number, less than -999 or more than +999, was specified in the Y-register.

"ALPHA DATA" Either the data file register, data file pointer value in register-Y, or the X-register contains alphanumeric data and therefore cannot accept arithmetical operations.

## STOIDF

**STOre X in Data File register**

**XROM 04,45**

**and increment pointer**

Details: The content of the X-register will be stored at the current location of the data file pointer in the working data file and the pointer will then be incremented by one.

Input: The X-register should contain the value to be stored in the data file register.
The data file pointer should specify the register into which the X-register is to be stored.

Output: The X-register content will be copied to the data file register.
The data file pointer will have been incremented by one.

Example:

| Prgm Step | Display | comments |
|---|---|---|
| 12 | 12_ | Value to set pointer to. |
| "DFTEST" | DFTEST | Name of the data file. |
| RSTDFX | 12.0000 | Reset "DFTEST" as the working file and sets pointer to DF:12. |
| 5 | 5_ | Value to be stored. |
| STOIDF | 5.0000 | Value stored in DF:12 and pointer incremented by one to point to DF:13. |
| RPTDF | 13.0000 | Recall pointer value. |
| STOIDF | 13.0000 | Value stored into DF:13 and pointer incremented. |
| RPTDF | 14.0000 | Recall pointer value |

Errors: Refer to the *HP-41 Owners Manuals* for details of displayed standard messages.

"NO WORKFL" There is no current or working data file selected. Use **RSTDF** to select a working file.

"END OF FL" The data file pointer value is greater than the size of the selected file and thus points to a non-existent register. Change the pointer to a value less than the size of the data file, or increase the file size with the **RESDF** function.

## X<>DF

**X exchange with Data**

**XROM 04,53**

**File register**

Details: The value contained in the X-register is exchanged with the data file register indicated by the current pointer value.

Input: The X-register should contain the value to be exchanged with that in the data file register.

Output: The X-register will contain the value previously in the data file register.
The data file register will contain the value previously in the X-register.
The pointer value will remain unchanged.

| Example: | Prgm Step | Display | comments |
|---|---|---|---|
| | 5 | 5_ | |
| | SPTDF | | Pointer to DF:5 in working file. |
| | 23 | 23_ | |
| | STODF | 23.0000 | Stores 23.0000 in DF:5 |
| | 8 | 8_ | |
| | X<>DF | 23.0000 | Data file and X-register values have been exchanged. |
| | X<>DF | 8.0000 | Values have been exchanged back again. |

Errors: Refer to the *HP-41 Owners Manuals* for details of displayed standard messages.

"END OF FL" The data file pointer contains a value greater than the size of the selected file and is therefore pointing to a non-existent register. Change the pointer to a value less than the size of the data file, or increase the file size with the **RESDF** function.

"NO WORKFL" There is no current or working data file selected. Use **RSTDF** to select a working file.

## X<>DFY

**X exchange with Data File register at Y**

**XROM 04,53**

Details: The data file pointer is set to the value in register-Y and the value in the X-register is exchanged with the data file register specified by this new pointer position.

Input: The Y-register should contain the new position for the data file pointer.
The X-register should contain the value to be exchanged with that in the data file register.

Output: The values contained in the X-register and the data file register indicated by the new pointer position will be exchanged.
The pointer position will be reset to the value in the Y-register.
The contents of the Y,Z,T and LASTX-registers will remain unchanged.

Example:

| Prgm Step | Display | comments |
|---|---|---|
| 25 | 25_ | |
| SPTDF | | Sets pointer to DF:25 in the working file. |
| 823 | 823_ | |
| STODF | 823.0000 | Stores 823 in DF:25 |
| X<>Y | 25.0000 | Recalls value to use for pointer. |
| 1923.89 | 1923.89_ | Value to exchange with DF:25. |
| X<>DFY | 823.0000 | Data file (pointer value in Y-reg) and X-reg values exchanged. |
| X<>DFY | 1923.8900 | Values exchanged back again. |

Errors: Refer to the *HP-41 Owners Manuals* for details of displayed standard messages.

"END OF FL" The data file pointer is positioned to a non-existent register past the end of the file. Change the pointer to a value less than the data file size, or increase the file size.

"NO WORKFL" No working data file is selected. Use **RSTDF** to select a working file.

"ALPHA DATA" Either the data file register or X-register contains alphanumeric data and cannot accept arithmetical operations.
The Y-register contains alphanumeric data that cannot be used as the new pointer position.

"NONEXISTENT" An illegal number, less than -999 or more than +999, was specified in the X-register.

# Section 5

# DATA FILE MOVEMENT

**Section 5** describes functions for working with data in groups of registers at one time. They can be functionally divided into three specific categories:

The first group operate on **all** main memory registers allocated to data storage (using the SIZE function) in one operation. These functions affect a group of the same number of registers within the RSU Data File:

| | |
|---|---|
| **EXREG** | EXchange REGisters |
| **RCLREG** | ReCaLl to REGisters |
| **STOREG** | STOre REGisters |

The second group operate on a specified quantity of main memory registers at one time. The user specifies that quantity by putting a value in the X-register and execution affects that same quantity of registers within the RSU Data File:

| | |
|---|---|
| **EXREGX** | EXchange REGisters by X |
| **RCLREGX** | ReCaLl REGisters by X |
| **STOREGX** | STOre REGisters by X |

The last group always operate on five registers at once, namely the X, Y, Z, T and LASTX stack registers.

| | |
|---|---|
| **EXST** | EXchange STack registers |
| **RCLST** | ReCaLl to STack registers |
| **STOST** | STOre STack registers |

In the following function descriptions, RSU-data file register numbers are indicated as "DF:xx". Main memory register numbers are indicated as either "R:xx" or "Reg:xx". For both types of registers, the number of a particular register is indicated by the value "xx".

**EXREG** **EXchange main &**

**XROM 04,15** **rsu REGisters**

Details: The content of all main memory data registers is exchanged with registers from the working RSU file. RSU registers are exchanged beginning at the current pointer position, working towards the direction of the file end.
If the pointer is initially positioned such that the number of main memory registers to exchange will exceed the size of the data file, then only a partial exchange will occur. Execution will be terminated after the last possible exchange is made and the X-register will be cleared to indicate this.
If the number of main memory registers exchanged exactly matches the number of RSU-registers from the pointer to the file end, then the X-register is also cleared.

Input: The pointer must be positioned to the first RSU register to be exchanged with main memory.

Output: The content of main memory and RSU registers from the working data file will be exchanged.
In the case of an exact or partial exchange of registers, the previous value of the X-register will be cleared.
The pointer is repositioned to the RSU register following the last one exchanged, or to the end of file where reached.

Errors: Refer to the *HP-41 Owners Manuals* for details of displayed standard messages.

"NO WORKFL" There is no working data file selected. Use **RSTDF** to select a working file.

"END OF FL" The pointer currently points to the end of the working file and would be attempting to exchange main memory registers with non-existent RSU registers.
Use **RSTDF** or **MPTDF** to reset or move the pointer within the working data file, or increase the file size with the **RESDF** function.

## RCLREG

**ReCaLl from rsu to**

**XROM 04,31**

**main memory REGisters**

Details: The content of RSU-registers is recalled, beginning at the current pointer location, and placed into main memory storage registers beginning at R:000.
Execution of RCLREG is terminated by reaching either the end of the data file, or the end of main memory storage registers.

Input: The data file pointer should point to the beginning of the RSU-registers to recall.

Output: The content of main memory storage registers will be replaced by data from the RSU file beginning at the pointer position.
If all main memory is filled before the end of the RSU-file was reached, then the pointer is positioned to the register following the last one transferred.

If only a partial transfer took place, because the file end was reached, then the pointer is positioned to the file end.

Example: This assumes main memory has been SIZEd to 80 data registers and a RSU-file has previously been created of size 100-registers.

| Prgm Step | Display | Comments |
|---|---|---|
| "DFTEST" | | |
| 80 | 80__ | Value to set pointer to. |
| RSTDFX | | Resets file & pointer to DF:80. |
| RCLREG | 0.0000 | Recalls from pointer at DF:80 towards end of file. Reg-X is cleared indicating only partial recall took place. |
| RPTDF | 100.0000 | Pointer now set to DF:80. |
| RCLREG | END OF FL | Cannot recall because pointer is already at file end. |

Errors: Refer to the *HP-41 Owners Manuals* for details of displayed standard messages.

"NO WORKFL" There is no working data file selected. Use **RSTDF** to select a working file or **CRDF** to create a new RSU data file.

"END OF FL" The data file pointer contains a value greater than the size of the selected file and is therefore pointing to a non-existent register. Reset the pointer to a value less than the size of the data file, or increase the file size.

## STOREG

**STOre main memory**

**XROM 04,48**

**REGisters into RSU file**

Details: The content of main memory registers is stored in the working RSU-file beginning at the current position of the data file pointer.
Execution is terminated when either all main memory has been stored or the end of the RSU-file is reached.

Input: The pointer should be positioned to the first of the RSU-registers into which main memory is to be stored.

Output: The working RSU-file contains a copy of the content of main memory storage registers.
Where the end of file was reached, only those main memory registers that could be accommodated are stored in the file. In this case, the pointer is positioned to the file end and the X-register cleared to indicate partial completion.

Where all main memory registers were stored, the pointer is positioned to the RSU-register following the last stored.

Example: This assumes main memory has been SIZEd to 80 data registers and a RSU-file has previously been created of size 100-registers.

| Prgm Step | Display | comments |
|---|---|---|
| "DFTEST" | | |
| RSTDF | | Resets working file. |
| STOREG | | Stores R:00 to R:79 in file. |
| RPTDF | 80.0000 | Pointer now set to 80. |
| STOREG | 0.0000 | Main memory is again stored in the file. But only R:00 to R:19 can be stored before end of file. Reg:X cleared to indicate this. |
| RPTDF | 100.0000 | Pointer now positioned to the end of file. |
| STOREG | END OF FL | Error indicates attempt to store into non-existent registers. |

Errors: Refer to the *HP-41 Owners Manuals* for details of displayed standard messages.

"NO WORKFL" There is no working data file selected. Use **RSTDF** to select a working file or **CRDF** to create a new RSU data file.

"END OF FL" An attempt was made to execute **STOREG** with the pointer positioned to the RSU-file end. Reset the pointer (**RSTDF**) to a value less than the size of the file, or increase the size with the **RESDF** function.

# EXREGX

**EXchange REGisters by X**

**XROM 04,16**

Details: The content of specified main memory storage registers is exchanged with that of registers from the working RSU-file beginning at the current position of the pointer.
If the pointer is initially positioned such that the number of main memory registers to exchange will exceed the size of the data file, then only a partial exchange will occur and execution will terminate after the last possible exchange is made. Partial exchanges will be indicated by a clearing of the X-register.
If the number of main memory registers exchanged exactly matches the number of RSU-registers from the pointer to the file end, then the X-register is also cleared.

Input: The X-register should contain the control word "*bbb.eee*" where "*bbb*" indicates the starting register, and "*eee*" the ending register.
If "*eee*" is greater than "*bbb*", then only register "*bbb*" is exchanged.
The pointer position indicates the first data file register to be exchanged.

Output: The main memory and RSU registers from the current working file will be exchanged.
In the case of an exact or partial exchange, the X-register will be cleared.
The RSU-pointer will be repositioned to the register following the last one exchanged. If the end of file is reached, the pointer will be positioned at the end.

Errors: Refer to the *HP-41 Owners Manuals* for details of displayed standard messages.

"NO WORKFL" There is no working data file selected. Use **RSTDF** to select a working file.

"END OF FL" The pointer currently points to the end of the working file and an attempt was made to exchange main memory registers with non-existent RSU registers.
Use **RSTDF** or **MPTDF** to reset or move the pointer within the working file, or increase the size with the **RESDF** function.

# RCLREGX

**ReCaLl REGisters by X**

XROM 04,32

Details: The content of specified main memory storage registers is replaced by that of RSU-registers beginning at the current pointer location.

Cautions: **Because main memory storage registers will be overwritten the user should ensure that wanted data is stored elsewhere.**

Input: The X-register should contain the control word "*bbb.eee*", where "*bbb*" and "*eee*" indicate respectively the beginning and end of the main memory registers to be replaced.
If "*eee*" is greater than "*bbb*", then only register "*bbb*" is replaced.
The RSU-pointer should be positioned to the first data register to be recalled to main memory.

Output: The content of specified main memory registers will be replaced by data from the RSU file.

Execution is halted by reaching the end of either the data file, or main memory storage registers.
If the complete operation is performed, then the pointer is positioned to the RSU-register following the last one recalled.
If the operation is terminated because the RSU-file end was reached, then the pointer is positioned at the end.

Example: **This assumes main memory has been SIZEd to 80 data registers and a RSU-file has previously been created of size 100-registers.**

| Prgm Step | Display | Comments |
|---|---|---|
| "DFTEST" | | |
| 0.020 | 0.020_ | Control word for begin & end. |
| RSTDF | | Resets working file. |
| RCLREGX | 0.0200 | Recalls from pointer towards file end. Placed in Main Memory R:00 to R:20. |
| RPTDF | 21.0000 | Pointer now set to DF:21. |

Errors: Refer to the *HP-41 Owners Manuals* for details of displayed standard messages.

"NO WORKFL" There is no working data file selected. Use **RSTDF** to select a working file or **CRDF** to create a new RSU data file.

"END OF FL" The data file pointer is positioned to the file end and therefore cannot perform the **RCLREGX**. Reset the pointer to a value less than the size of the data file, or increase the file size with the **RESDF** function.

"ALPHA DATA" The X-register contains alphanumeric data. Change to the correct control word format.

# STOREGX

**STOre REGisters by X**

**XROM 04,49**

Details: The content of specified main memory registers is stored in the working RSU-file beginning at the current position of the data file pointer.
Execution is terminated when either all specified main memory has been stored, or the end of the data file is reached.

Input: The X-register should contain the control word "*bbb.eee*", where "*bbb*" and "*eee*" indicate respectively the beginning and end of the main memory registers to be stored into the RSU-file.
If "*eee*" is greater than "*bbb*", then only register "*bbb*" is stored.
The RSU-pointer should be positioned to the first RSU-data register to be used for storage.

Output: The contents of specified main memory registers will be stored into the RSU-file.

Execution is terminated either at the end of the data file, or when all specified registers are stored.
If the complete operation is performed, then the pointer is positioned to the register following the last one stored.
When terminated because the end of the RSU-file was reached, the pointer will be relocated to the end and the X-register will be cleared.

Example:

| Prgm Step | Display | Comments |
|---|---|---|
| "DFTEST" | DFTEST | File to declare working file. |
| 50 | 50_ | Value for pointer |
| RSTDFX | 50.0000 | Resets working file & pointer to DF:50. |
| 0.010 | 0.010_ | Specifies control word (begin at R:00 & end at R:10). |
| STOREGX | | Stores R:00 to R:10 in file beginning at DF:50. |
| RPTDF | 60.0000 | Pointer is now set to DF:60. |
| 15 | 15_ | |
| MPTDF | 15.0000 | Move pointer by 15 to DF:75 |
| 33.035 | 33.035_ | specify begin (R:33) & end (R:35) |
| STOREGX | 33.0350 | Store R:33 to R:35 in the RSU-file from DF:75 onward. |
| RPTDF | 78.0000 | Pointer now positioned to DF:78. |

Errors: Refer to the *HP-41 Owners Manuals* for details of displayed standard messages.

"NO WORKFL" There is no working data file selected. Use **RSTDF** to select a working file or **CRDF** to create a new RSU data file.

"END OF FL" An attempt was made to execute **STOREGX** with the pointer positioned to the end of the RSU-data file. Reset the pointer (**RSTDF**) to a value less than the size of the data file, or increase the file size (**RESDF**).

| | |
|---|---|
| "NONEXISTENT" | The control word in the X-register does not correspond to the number of main memory registers available. Correct the value, or reSIZE the number of data storage registers. |
| "ALPHA DATA" | The X-register does not contain numerical data. Change to a number in the control word format. |
| "END OF FL" | The data file pointer contains a value greater than the size of the selected file and is therefore pointing to a non-existent register. Reset the pointer to a value less than the size of the data file, or increase the file size with the **RESDF** function. |

# EXST

**EXchange STack registers**

**XROM 04,17**

Details: The content of stack registers X,Y,Z,T and LASTX are exchanged with five RSU-registers beginning at the current pointer position. The stack is exchanged progressively from Reg:T at the current pointer position, to Reg:LASTX at the pointer position **+4**.

Input: The data file pointer should be positioned to the first of the RSU-registers to be exchanged. To prevent an "END OF FL" error occurring, this must be at least five registers from the file end.

Output: The contents of the stack and five registers in the working RSU-file are exchanged.
The pointer is repositioned to the first RSU-register following the last one exchanged.

Errors: Refer to the *HP-41 Owners Manuals* for details of displayed standard messages.

"NO WORKFL" Use **RSTDF** to select a working file or **CRDF** to create a new RSU data file.

"END OF FL" The pointer is positioned less than five registers from the file end and thus points to non-existent registers.
Reset the pointer to at least five less than the file size, or increase the size.

# RCLST

**ReCaLl to STack registers**

XROM 04,33

Details: The contents of five RSU-registers beginning at the current pointer position are recalled to the stack registers X,Y,Z,T and LASTX.

Input: The pointer should be positioned to the first register to recall. Must be at least five registers from the end to prevent an END OF FL error.

Output: The stack register contents will be replaced by data from the RSU-file. Reg:T corresponds to the DF:register indicated by the pointer and the LASTX-register corresponds to the pointer **+4**. The pointer is repositioned to the register following the last recalled.

Errors: Refer to the *HP-41 Owners Manuals* for details of displayed standard messages.

"NO WORKFL" Use **RSTDF** to select a working file or **CRDF** to create a new RSU data file.

"END OF FL" The pointer is positioned less than five registers from the end of the file. Reset the pointer to at least five less than the file size, or increase the size.

# STOST

**STOre STack registers**

**XROM 04,50**

**into rsu data block**

Details: The contents of the stack registers, X,Y,Z,T and LASTX are stored into the working data file beginning at the current pointer position. The stack registers are stored progressively from the T-register at the current pointer position, to the LASTX register at the pointer position **+4**.

Input: The pointer should be positioned to the first RSU-register to be used for storing the stack.

Output: The stack contents will be stored in the RSU-file beginning at the previous pointer position and the pointer relocated to the first data file register following the stored registers.

Example:

| Prgm Step | Display | Comments |
|---|---|---|
| 5 | 5_ | |
| SPTDF | 5.0000 | Sets pointer to DF:5 |
| STOST | 5.0000 | Stack is stored beginning at DF:5 |
| CHS | -5.0000 | |
| MPTDF | -5.0000 | pointer is moved by 5 backwards |
| RCLIDF | "T-Reg" | displays stored content of T-register and increments pointer to DF:06 |
| RCLIDF | "Z-Reg" | displays stored Z-register and increments pointer to DF:07 |
| RCLIDF | "Y-Reg" | Displays stored Y-register and increments pointer to DF:08 |
| RCLIDF | "X-Reg" | Displays stored X-register and increments pointer to DF:09 |
| RCLIDF | "LASTX-Reg" | Displays stored LASTX-register and increments pointer to DF:10 |
| RPTDF | 10.0000 | Recalls current pointer position. |

Errors: Refer to the *HP-41 Owners Manuals* for details of displayed standard messages.

"NO WORKFL" There is no current or working data file selected. Use **RSTDF** to select a working file or **CRDF** to create a new data file.

"END OF FL" The pointer is positioned less than five registers from the end of the specified file and thus points to non-existent registers. Reset the pointer to at least 5-data registers from the end, or increase the size with **RESDF**.

# Section 6

# POINTER OPERATIONS

Most of the functions contained in the **ES-41 DATABASE** module require the use of a pointer to indicate the particular RSU-register, or the beginning of a block of RSU-registers, on which the chosen operation is to be performed.

**Section 6** describes those **ES-41 DATABASE** functions that allow the user to manipulate the data file pointer within the current working file.

| | |
|---|---|
| **DPTDF** | Decrement PoinTer Data File |
| **IPTDF** | Increment PoinTer Data File |
| **MPTDF** | Move PoinTer Data File |
| **RPTDF** | Recall PoinTer Data File |
| **SPTDF** | Set PoinTer Data File |
| **X=PT?** | X-register equals PoinTer ? |

In the following function descriptions, RSU-data file register numbers are indicated as "DF:xx". Main memory register numbers are indicated as either "R:xx" or "Reg:xx". For both types of registers, the number of a particular register is indicated by the value "xx".

# DPTDF — Decrement PoinTer Data File

## XROM 04,12

Details: The value of the data file pointer is decremented by one.

Cautions: **Where the pointer is already set to DF:00 (i.e. set to the beginning of the file), the pointer is not decremented. No error message is reported.**

Input: None.

Output: The pointer value is decremented by one.

Example:

| Prgm step | display | comments |
|---|---|---|
| 15 | 15_ | |
| SPTDF | 15.0000 | Set pointer to DF:15. |
| DPTDF | 15.0000 | Decrement the pointer by one. |
| RPTDF | 14.0000 | Recall decremented pointer. |

Errors: Refer to the *HP-41 Owners Manuals* for details of displayed standard messages.

"NO WORKFL" There is no working data file selected. Use **RSTDF** to select a working file or **CRDF** to create a new RSU data file.

Details: The data file pointer is incremented by one.

Cautions: **Where the pointer is already set to the last data file register, executing IPTDF will position the pointer at the end of file, but no error message will be reported. Where the pointer is already positioned to the end of file, the "END OF FL" error will be reported.**

Input: None.

Output: The pointer position will be incremented by one.

Errors: Refer to the *HP-41 Owners Manuals* for details of displayed standard messages.

| | |
|---|---|
| "END OF FL" | The data file pointer is positioned to a non-existent register past the end of the file. Change the pointer to a value less than the data file size, or increase the file size. |
| "NO WORKFL" | There is no working data file selected. Use **RSTDF** to select a working file or **CRDF** to create a new RSU data file. |

# MPTDF

**Move PoinTer Data File**

**XROM 04,25**

Details: The data file pointer is moved by the number of registers specified in the X-register.
The sign of the X-register value determines the direction of pointer movement.

Cautions: **Where the pointer is already set to DF:00, no movement will occur and no error message will be reported.**

EOF?

Input: The X-register should contain the value by which the pointer is to be moved. A positive value will cause the pointer to be incremented, while a negative value will cause the pointer to be decremented.

Output: The Data File pointer will be set to its new value.

Example:

| Prgm step | display | comments |
|---|---|---|
| "TESTDF" | | File to declare working file. |
| RSTDF | | Resets working file and sets pointer to DF:00. |
| 5 | 5_ | |
| CHS | -5_ | |
| MPTDF | -5.0000 | Move pointer by -5. |
| RPTDF | 0.0000 | Recall pointer still at DF:00. |
| 10 | 10_ | |
| MPTDF | 10.0000 | Move pointer by +10 |
| RPTDF | 10.0000 | Recall pointer, now at DF:10. |
| 2 | 2_ | |
| CHS | -2_ | |
| MPTDF | -2.0000 | Move pointer by -2. |
| RPTDF | 8.0000 | Recall new value of pointer set at DF:8. |

Errors: Refer to the *HP-41 Owners Manuals* for details of displayed standard messages.

"ALPHA DATA" The X-register contains alphanumeric data and therefore cannot be used as the pointer value.

"END OF FL" The data file pointer is positioned to a non-existent register past the end of the file. Change the pointer to a value less than the data file size, or increase the file size.

"NONEXISTENT" An illegal number, less than -999 or more than +999, was specified in the X-register.

"NO WORKFL" There is no working data file selected. Use **RSTDF** to select a working file or **CRDF** to create a new RSU data file.

# RPTDF

**Recall Pointer of Data File**

**XROM 04,38**

Details: The current value of the data file pointer is recalled to the X-register.

Input: None

Output: The value in the X-register is replaced by the current position of the data file pointer within the working file.

Example:

| Prgm step | display | comments |
|---|---|---|
| 15 | 15_ | |
| SPTDF | 15.0000 | Set Data File pointer to DF:15 |
| CLX | 0.0000 | Clear the X-register |
| RPTDF | 15.0000 | Recall current pointer value. |

Errors: Refer to the *HP-41 Owners Manuals* for details of displayed standard messages.

"NO WORKFL" There is no working data file selected. Use **RSTDF** to select a working file or **CRDF** to create a new RSU data file.

# SPTDF

**Set PoinTer of DataFile**

**XROM 04,42**

**Details:** The data file pointer within the working file is set to the value specified in the X-register.

**Input:** The X-register should contain the data file register number to which the pointer is to be set.

**Output:** The data file pointer will be set to its new value.

Example:

| Prgm step | Display | Comment |
|---|---|---|
| "DFTEST" | | File to declare working file. |
| RSTDF | 0.0000 | Reset working file to DF:00 |
| 15 | 15_ | Position to set pointer to. |
| SPTDF | 15.0000 | Sets pointer position to DF:15 |
| CLX | | Clears Reg:X |
| RPTDF | 15.0000 | Recall new pointer position. |

Errors: Refer to the *HP-41 Owners Manuals* for details of displayed standard messages.

| | |
|---|---|
| "NO WORKFL" | There is no working data file selected. Use **RSTDF** to select a working file or **CRDF** to create a new RSU data file. |
| "NONEXISTENT" | An illegal number, less than -999 or more than +999, was specified in the X-register. |
| "END OF FL" | The data file pointer is positioned to a non-existent register past the end of the file. Change the pointer to a value less than the data file size, or increase the file size. |

# X=PT?

**X-register = df PoinTer ?**

**XROM 04,61**

**Details:** Checks whether or not the value of the X-register is equal to the current data file pointer position.

**Cautions:** **A comparison is made for both alphanumeric and numeric data. No error message is reported if Reg:X contains an alphabetic character.**

**Input:** The X-register should contain the value to compare the pointer position with.

**Output:** Output depends upon method of execution:

**Keyboard Execution:** causes a displayed YES/NO answer:

| Display | Meaning |
|---|---|
| YES | The values are equal |
| NO | The values are not equal |

**Program Execution:** causes no displayed answer:

The function follows the "DO IF TRUE" rule

Example: The following program illustrates the use of **X=PT?** to check whether or not the end of file has been reached while prompting for and inserting values into successive data file registers.

```
01*LBL'INPUT'
02 10          size of RSU-data file in registers.
03 'TEST'      name to give to RSU-data file.
04 CRDF        creates a data file of 10 registers and with
               name 'TEST'. Also declares 'TEST' to be
               the working file and sets pointer to DF:00.
05*LBL 01
06 'VALUE ?'
07 PROMPT      prompts for input of a value
08 STOIDF      stores value and increments the data file
               pointer by one register.
09 SZ?DF       Checks size of the working file by recalling
               value to X-register.
10 X=PT?       Is file size equal to current pointer
               position ?
11 GTO 02      If yes, go to LBL 02.
12 GTO 01      If no, go to LBL 01 to input next value.
13 '** DONE **'
14 AVIEW       Display message to indicate end of file.
15 END
```

Errors: Refer to the *HP-41 Owners Manuals* for details of displayed standard messages.

"NO WORKFL" There is no working data file selected. Use **RSTDF** to select a working file or **CRDF** to create a new RSU data file.

# Section 7

# DATA FILE COMPARISON

**Section 7** describes functions concerned with search and comparison. The functions are divided into two subgroups:

## The Search Functions:

| | |
|---|---|
| **FIND** | FIND target value |
| **FINDI** | FIND target value Inverse |

These permit the user to search all registers in the working data file for equality with a numeric or alphanumeric value in the X-register.

In addition to defining the direction of search through the file, the user may define a "*step size*" specifying the number of registers to be skipped between register comparisons.

If the X-register search value is found, the pointer is repositioned to the RSU-register containing that value.

**Keyboard Execution:**

When executed from the keyboard, these functions display either a **YES** or **NO** answer depending upon the outcome of the test:

**Program Execution:**

Under program control, there is no displayed result, but the program instruction following the search operation will be skipped or executed depending upon whether or not the search value was found. For example, **FIND** behaves in the following manner:

| | |
|---|---|
| *Value Found:* | Executes the next program instruction. |
| *Value Not Found:* | Skips the next program instruction. |

Note: The FINDI function performs the inverse of these results.

Specifying the step size is particularly useful when an application uses a multi-field record spread over more than one register. By this, the user can specify, for example, that only field two (perhaps containing a surname) be searched in each record entry.

## The Comparison Functions:

| | |
|---|---|
| **X=DF?** | X = Data File register ? |
| **X<>DF?** | X <> Data File register ? |
| **X<DF?** | X < Data File register ? |
| **X<=DF?** | X <= Data File register ? |
| **X>DF?** | X > Data File register ? |
| **X>=DF?** | X >= Data File register ? |

These functions allow the user to compare the numeric or alphanumeric value specified in the X-register against the content of a specific RSU-register indicated by the data file pointer. Note that when numeric and alphanumeric values are compared, alphanumeric values are always considered GREATER THAN numeric values.

Comparison functions perform tests in the manner of the standard HP-41 tests such as **X=Y?**, **X<=Y?**, etc.

**Under Keyboard Execution:** a **YES** or **NO** answer is displayed depending upon the test result:

| | |
|---|---|
| **YES** | **The comparison test result was TRUE.** |
| **NO** | **The comparison test result was FALSE.** |

**Under Program Execution:** the **"DO IF TRUE"** rule is followed depending upon the particular test result:

| **IF test TRUE** | **IF test FALSE** |
|---|---|
| PERFORM next step | SKIP next step |

# FIND

**FIND target value**

**XROM 04,18**

Details: The working RSU-data file is searched for a numeric or alphanumeric value specified in the X-register. If found, the pointer is repositioned to the register containing the value.
Both the direction of search through the data file and the step size with which the search takes place may be controlled by the user.
Searching begins from the current pointer position and terminates when either, the top or end of the working file is reached, or the target value has been found.

Cautions: **Where the pointer is already set to the end of file, execution of FIND will not take place. The pointer should be decremented by at least one before FIND will operate.**

Input: The X-register should contain the search value, which may be either numeric or alphanumeric.
The Y-register should contain a value indicating the direction and the step size. A negative value performs searching from the current pointer position towards the top of file. A positive value performs searching from the current pointer position towards the end of file.
The value itself specifies the quantity of registers to be skipped during the search.

Output: If the search value is found, the pointer is repositioned to the RSU-register containing that value.
Further output depends upon execution method. (*See also descriptions on pages 99 and 100.*):

**Keyboard Execution:** causes a YES/NO displayed answer.

| | |
|---|---|
| **YES** | The X-register search value WAS found. |
| **NO** | The X-register search value WAS NOT found. |

**Program Execution:** causes no answer to be displayed.

| | |
|---|---|
| *Value Found:* | Executes the following program instruction. |
| *Value Not Found:* | Skips the following program instruction. |

Errors: Refer to the *HP-41 Owners Manuals* for details of displayed standard messages.

| | |
|---|---|
| "NO WORKFL" | There is no working data file selected. Use **RSTDF** to select a working file or **CRDF** to create a new RSU data file. |
| "NONEXISTENT" | An illegal number, less than -999 or more than +999, was specified in the Y-register. |
| "ALPHA DATA" | The Y-register contains alphanumeric data. |
| "DATA ERROR" | The step size in the Y-register is zero. Change the value and retry. |

# FINDI

**FIND target value Inverse**

**XROM 04,19**

Details: The working RSU-data file is searched for a numeric or alphanumeric value specified in the X-register. The action taken by the function is the inverse of the FIND function.
If found, the pointer is repositioned to the register containing the value.
Both the direction of search through the data file and the step size with which the search takes place may be controlled by the user.
Searching begins from the current pointer position and terminates when either the top or end of the working file is reached, or the target value has been found.

Cautions: **Where the pointer is already set to the end of file, execution of FINDI will not take place. The pointer should be decremented by at least one before FINDI will operate.**

Input: The X-register should contain the search value, which may be either numeric or alphanumeric.
The Y-register should contain a value indicating the direction and the step size. A negative value performs searching from the current pointer position towards the top of file. A positive value performs searching from the current pointer position towards the end of file.
The value itself specifies the quantity of registers to be skipped during the search.

Output: If the search value is found, the pointer is repositioned to the RSU-register containing that value.
Further output depends upon execution method. (*See also descriptions on pages 99 and 100.*):

**Keyboard Execution:** causes a YES/NO displayed answer.

| | |
|---|---|
| **NO** | The X-register search value WAS found. |
| **YES** | The X-register search value WAS NOT found. |

**Program Execution:** causes no answer to be displayed.

| | |
|---|---|
| *Value Not Found:* | Executes the following program instruction. |
| *Value Found:* | Skips the following program instruction. |

Errors: Refer to the *HP-41 Owners Manuals* for details of displayed standard messages.

| | |
|---|---|
| "NO WORKFL" | There is no working data file selected. Use **RSTDF** to select a working file or **CRDF** to create a new RSU data file. |
| "NONEXISTENT" | An illegal number, less than -999 or more than +999, was specified in the Y-register. |
| "ALPHA DATA" | The Y-register contains alphanumeric data. |
| "DATA ERROR" | The step size in the Y-register is zero. Change the value and retry. |

## X=DF?

**X-register EQUAL to Data File register ?**

**XROM 04,55**

Details: A comparison for equality is made between a numeric or alphanumeric value in the X-register and the value in the data file register at the current pointer position.

Input: The pointer should be positioned to the data file register whose value is to be compared.
The X-register should contain the value to be compared against the data file register.

Output: Output depends upon execution method. (*See also page 101 for a fuller description*):

**Keyboard Execution:**

**YES** X- and data file register values ARE EQUAL.
**NO** X- and data file registers ARE UNEQUAL.

**Program Execution:**
Program flow is controlled according to the test result and follows a "**DO IF TRUE**" rule:

| **Values EQUAL** | **Values UNEQUAL** |
|---|---|
| Perform Next Step | Skip Next Step |

Example:

| Keystrokes | display | comments |
|---|---|---|
| 25 | 25_ | |
| SPTDF | 25.0000 | Set pointer to DF:25. |
| 555 | 555_ | Value to store. |
| STODF | 555.0000 | Store 555 in DF:25. |
| 5 | 5_ | Value to compare. |
| X=DF? | NO | Values are not equal. |
| RCLDF | 555.0000 | Make X-register equal to DF:25. |
| X=DF? | YES | Values are indeed equal. |

Errors: Refer to the *HP-41 Owners Manuals* for details of displayed standard messages.

| | |
|---|---|
| "END OF FL" | The data file pointer is positioned to a non-existent register past the end of the file. Change the pointer to a value less than the data file size, or increase the file size. |
| "NO WORKFL" | There is no working data file selected. Use **RSTDF** to select a working file or **CRDF** to create a new RSU data file. |

## X<>DF?

**X-register NOT EQUAL to Data File register ?**

**XROM 04,56**

Details: The X-register value is checked for inequality with the value in the data file register at the current pointer position. Comparison is valid for both numerical and alphanumeric data.

Input: The X-register should contain the value to be compared against the RSU-register.
The pointer should be positioned at the RSU-register.

Output: Output depends upon execution method. (*See also page 101 for a fuller description*):

**Keyboard Execution:**

| | |
|---|---|
| **YES** | **X- and data file registers ARE UNEQUAL.** |
| **NO** | **X- and data file register values ARE EQUAL.** |

**Program Execution:**
Program flow is controlled according to the test result and follows a "**DO IF TRUE**" rule:

| **Values UNEQUAL** | **Values EQUAL** |
|---|---|
| Perform Next Step | Skip Next Step |

Example:

| keystrokes | display | comments |
|---|---|---|
| 25 | 25_ | |
| SPTDF | 25.0000 | Sets pointer to DF:25 |
| STODF | 25.0000 | Store 25.0000 in DF:25 |
| X<>DF? | NO | Values are NOT unequal |
| CLX | 0.0000 | |
| X<>DF? | YES | They ARE now unequal |

Errors: Refer to the *HP-41 Owners Manuals* for details of displayed standard messages.

"END OF FL" The data file pointer is positioned to a non-existent register past the end of

| | |
|---|---|
| | the file. Change the pointer to a value less than the data file size, or increase the file size. |
| "NO WORKFL" | There is no working data file selected. Use **RSTDF** to select a working file or **CRDF** to create a new RSU data file. |

## X<DF?

**X-register LESS THAN Data File register ?**

**XROM 04,57**

Details: The numeric or alphanumeric value in the X-register is compared for being less than the value in the data file register at the current pointer position.

Input: The X-register should contain the value to be compared against the RSU-register.

Output: Output depends upon execution method. (*See also page 101 for a fuller description*):

**Keyboard Execution:**

**YES** Reg:X value LESS THAN data file register.

**NO** Reg:X value EQUAL or GREATER THAN data file register.

**Program Execution:**

Program flow is controlled according to the test result and follows a "**DO IF TRUE**" rule:

| **LESS THAN** | **GREATER or EQUAL** |
|---|---|
| Perform Next Step | Skip Next Step |

Example:

| key steps | display | comments |
|---|---|---|
| 25 | 25_ | |
| SPTDF | 25.0000 | Sets pointer to DF:25. |
| STODF | 25.0000 | Stores 25 in DF:25. |
| X<DF? | NO | Reg:X is NOT smaller. |
| CLX | 0.0000 | Clears Reg:X. |
| X<DF? | YES | Reg:X IS now smaller. |

Errors: Refer to the *HP-41 Owners Manuals* for details of displayed standard messages.

"END OF FL" The data file pointer is positioned to a non-existent register past the end of the file. Change the pointer to a value

| | |
|---|---|
| | less than the data file size, or increase the file size. |
| "NO WORKFL" | There is no working data file selected. Use **RSTDF** to select a working file or **CRDF** to create a new RSU data file. |

## X<=DF? X-register LESS THAN OR EQUAL

**XROM 04,58** **to Data File register?**

Details: The numeric or alphanumeric value in the X-register is compared for being equal or less than the value of the data file register at the current pointer position.

Input: The X-register should contain the value to compare against the RSU-register.
The pointer should be positioned to the RSU-register to be compared.

Output: Output depends upon execution method. (*See also page 101 for a fuller description*):

**Keyboard Execution:**

**YES** Reg:X IS EQUAL TO or LESS THAN data file register.

**NO** Reg:X IS GREATER THAN data file register.

**Program Execution:**
Program flow is controlled according to the test result and follows a "**DO IF TRUE**" rule:

| **LESS THAN or EQUAL** | **Value GREATER** |
|---|---|
| Perform Next Step | Skip Next Step |

Example:

| key steps | display | comments |
|---|---|---|
| 25 | 25_ | |
| SPTDF | 25.0000 | Sets pointer to DF:25 |
| STODF | 25.0000 | Store 25 in DF:25 |
| X<=DF? | YES | Reg:X EQUAL or LESS |
| 30 | 30_ | Change Reg:X value |
| X<=DF? | NO | Reg:X now larger than DF:25. |

Errors: Refer to the *HP-41 Owners Manuals* for details of displayed standard messages.

"END OF FL" The data file pointer is positioned to a non-existent register past the end of

| | |
|---|---|
| | the file. Change the pointer to a value less than the data file size, or increase the file size. |
| "NO WORKFL" | There is no working data file selected. Use **RSTDF** to select a working file or **CRDF** to create a new RSU data file. |

## X>DF?

**X-register GREATER THAN**

**XROM 04,59**

**Data File register ?**

Details: The numeric or alphanumeric value in the X-register is compared for being greater than the value of the data file register at the current pointer position.

Input: The X-register should contain the value to compare against the RSU-register.
The pointer should be positioned to the RSU-register to be compared.

Output: Output depends upon execution method. (*See also page 101 for a fuller description*):

**Keyboard Execution:**

| | |
|---|---|
| **YES** | **Reg:X IS GREATER THAN data file register.** |
| **NO** | **Reg:X IS EQUAL TO or LESS THAN data file register.** |

**Program Execution:**
Program flow is controlled according to the test result and follows a "**DO IF TRUE**" rule:

| **Value GREATER** | **LESS THAN or EQUAL** |
|---|---|
| Perform Next Step | Skip Next Step |

Example:

| key steps | display | comments |
|---|---|---|
| 25 | 25_ | |
| SPTDF | 25.0000 | Set pointer to DF:25 |
| [ALPHA] | | Press ALPHA-mode key |
| CLA | | Clear ALPHA |
| "ABC" | ABC_ | Key-in string: ABC |
| ASTO X | ABC | Also store this in Reg:X |
| [ALPHA] | ABC | Press ALPHA-mode key again. |
| STODF | ABC | Store "ABC" into DF:25 |
| X>DF? | NO | Reg:X NOT GREATER than DF:25 |
| 55 | 55_ | |

| | | |
|---|---|---|
| X<>DF | ABC | Exchange Reg:X & DF:25 values |
| X>DF? | YES | Reg:X now tests as GREATER THAN DF:25 value because Alpha-data is always considered greater than numeric data. |

Errors: Refer to the *HP-41 Owners Manuals* for details of displayed standard messages.

"END OF FL" The data file pointer is positioned to a non-existent register past the end of the file. Change the pointer to a value less than the data file size, or increase the file size.

"NO WORKFL" There is no working data file selected. Use **RSTDF** to select a working file or **CRDF** to create a new RSU data file.

## X>=DF? X-register GREATER THAN or EQUAL

**XROM 04,60** **to Data File register ?**

Details: The numeric or alphanumeric value in the X-register is compared for being equal or greater than the value in the data file register at the current pointer position.

Input: The X-register should contain the value to compare against the RSU-register.
The pointer should be positioned to the RSU-register to be compared.

Output: Output depends upon execution method. (*See also page 101 for a fuller description*):

**Keyboard Execution:**

**YES** Reg:X IS GREATER THAN or EQUAL TO the data file register.

**NO** Reg:X IS LESS THAN the data file register.

**Program Execution:**
Program flow is controlled according to the test result and follows a "**DO IF TRUE**" rule:

| GREATER or EQUAL | Value LESS THAN |
|---|---|
| Perform Next Step | Skip Next Step |

Example:

| key steps | display | comments |
|---|---|---|
| 25 | 25_ | |
| SPTDF | 25.0000 | Set pointer to DF:25 |
| [ALPHA] | | Press ALPHA-mode key |
| CLA | | Clear ALPHA |
| "ABC" | ABC_ | Key in string: ABC |
| ASTO X | ABC | Also store this in Reg:X |
| [ALPHA] | ABC | Press ALPHA-mode key again. |
| STODF | ABC | Store "ABC" into DF:25 |
| X>=DF? | YES | Reg:X EQUAL to or GREATER THAN value in DF:25. |
| 23 | 23_ | |

| | | |
|---|---|---|
| X>=DF? | NO | Numeric data is always considered LESS THAN alpha data, therefore test is false. |
| [ALPHA] | ABC | Press ALPHA-mode key. |
| "ABD" | ABD_ | Key in string: ABD |
| ASTO X | | |
| [ALPHA] | ABD | Press ALPHA-mode key again. |
| X=>DF? | YES | String in Reg:X ("ABD") tests GREATER THAN "ABC" in DF:25 |

Errors: Refer to the *HP-41 Owners Manuals* for details of displayed standard messages.

"END OF FL" The data file pointer is positioned to a non-existent register past the end of the file. Change the pointer to a value less than the data file size, or increase the file size.

"NO WORKFL" There is no working data file selected. Use **RSTDF** to select a working file or **CRDF** to create a new RSU data file.

# Section 8

# EXTENDED MEMORY FUNCTIONS

Extended memory is available to users of the HP-41-CX model, or to users of the HP-41-C or -CV models with the addition of the plug-in HP82180 Extended Functions and HP82181 Extended Memory Modules. Extended Memory Modules should not be confused with the Single or Quad memory modules that are available to expand the main memory of the basic HP-41C model.

Extended memory (EM) is a special area of HP-41 memory similar to the electronic or RAM disc on desktop computers. Although EM behaves as an on-line storage area for data and programs, (but cannot be used to run programs out of), it should not be used for permanent storage as the content will be lost if the power source fails, or the machine resets.

The amount of EM available to the HP-41 user depends upon the number of additional Extended Memory modules plugged in. The basic HP-41CX, or the HP-41C/CV with Extended Functions Module, contain 124 registers of user addressable EM. Up to two EM-Modules may be plugged in, each of 238 registers, giving a total of 600 EM-registers.[1]

The **ES-41 DATABASE** Module provides four additional functions enabling the RSU user to interact with EM.

| | |
|---|---|
| **CLEM** | CLear Extended Memory |
| **EXEM** | EXchange Extended Memory |
| **RCLEM** | ReCaLl Extended Memory |
| **STOEM** | STOre Extended Memory |

HP-41 Main Memory is not affected by these functions.

---

[1] In addition, there are some 8 registers used by the operating system itself.

# CLEM

**CLear Extended Memory**

**XROM 04,02**

Details: The entire contents of all available Extended Memory is cleared of all data and programs. The content of all main memory is unaffected by CLEM.

Cautions: **Be very cautious about using the CLEM function as it performs a memory lost for the entire Extended Memory. Once cleared, there is absolutely no way to recover the data!**

Input: None.

Output: The entire available Extended Memory is cleared of all data and programs.

Errors: None.

## EXEM — EXchange Extended Memory

**XROM 04,14**

**Details:** This function exchanges the entire contents of EM with the specified RSU-file of type "E" (Extended Memory).
Any number of Extended Memory images may be saved under different type "E" file names in the RSU; dependent upon the number of RSU-data blocks and EM currently configured.
The EM registers used by the HP-41 operating system are stored along with the type "E" file. Therefore, for the maximum possible EM, a total of 600 + 8 RSU-registers will be used.

**Input:** The Alpha register should contain the 1 to 7 character name of the RSU-file.

**Output:** The entire Extended Memory and the contents of the named RSU-file are swapped.
For the maximum number of 608 EM registers, the exchange operation will take about 15 seconds.

**Errors:** Refer to the *HP-41 Owners Manuals* for details of displayed standard messages.

"FL TYP ERR" The file specified in Alpha is not an type "E" (Extended Memory) file. Use FLDIR to check that the file exists.

"CONFIG ERR" The number of EM modules currently plugged into the HP-41 is not the same as when the type "E" file was created. Plug in, or unplug EM modules to restore the EM configuration used at creation time. Use FLDIR to determine the EM modules/registers needed.

"NOT A FILE" The file named in the Alpha-register does not currently exist.

| | |
|---|---|
| "NO EXT MEM" | The current HP-41 configuration does not include an Extended Functions module. Replace this module. |
| "NAME ERROR" | An illegal file name was specified in the Alpha-register. Only file names of 1 to 7 characters are allowed. |
| "NO FL NAME" | No file name was specified in the Alpha-register. |

## RCLEM — ReCaLl Extended Memory

**XROM 04,29**

**Details:** The content of the specified type "E" (Extended Memory) RSU-file is recalled to, and replaces the content of all Extended Memory registers.
The current EM module configuration should be identical to that at file creation time.

**Input:** The alpha register should contain the name of the type "E" file to be recalled.

**Output:** The content of EM is replaced by that of the type "E" file.

**Errors:** Refer to the *HP-41 Owners Manuals* for details of displayed standard messages.

| | |
|---|---|
| "FL TYP ERR" | The file specified in the Alpha is not an "E" type file. Use FLDIR to check that the file exists. |
| "CONFIG ERR" | The current configuration of EM-modules differs from that at the time the "E" type file was created. Plug in or unplug Extended Memory modules to restore the correct configuration. |
| "NOT A FILE" | The file named in the Alpha-register does not currently exist. |
| "NO EXT MEM" | The current HP-41 configuration does not include an Extended Functions module. Replace this module. |
| "NAME ERROR" | An illegal file name was specified in the Alpha-register. Only file names of 1 to 7 characters are allowed. |
| "NO FL NAME" | A file name was not specified in the Alpha-register. |

## STOEM

**STOre Extended Memory into rsu block.**

**XROM 04,46**

Details: The entire content of Extended Memory is stored in a type "E" (Extended Memory) file in a RSU-data block.
A file of the size necessary to store the entire, configured EM is automatically created. Where only the basic 41CX, or 41C/CV with Extended Function module is plugged in, the size is set to 126. For each additional EM-module, the type "E" file size is increased by 241 up to a maximum of 608 registers.[2]

Cautions: **A unique file name must be specified in Alpha.**

Input: The Alpha-register should contain a name for the new file of between 1 and 7 characters in length.

Output: A type "E" file with the specified name will be created in the first RSU data block with enough room. The file's length will match the currently configured EM modules.

Errors: Refer to the *HP-41 Owners Manuals* for details of displayed standard messages.

"DUP FL NAME" The specified file name already exists.

"NAME ERROR" An illegal file name was specified in the Alpha-register. File names of 1 to 7 characters are allowed.

"NO FL NAME" No file name was specified in the Alpha-register.

"NO ROOM" The number of RSU-registers needed to store the type "E" file is greater than the maximum free registers

[2] These numbers include those registers used by the 41's operating system. Only a maximum of 600 user addressable registers are available.

| | |
|---|---|
| | remaining in any one RSU-data block. Delete a file to free additional RSU-registers or initialise a new data block. |
| "NO RSU PAGE" | No RSU-block is currently enabled. Check the RSU-switches. RSU-blocks are either not initialised, or have been initialised for program, not data storage. |

# Section 9

## MAIN MEMORY FUNCTIONS

**Section 9** describes functions for interacting with the entire contents of the HP-41's main memory.

| | |
|---|---|
| **EXALL** | EXchange ALL main memory |
| **RCLALL** | ReCaLl ALL main memory |
| **STOALL** | STOre ALL main memory |

These functions now make it possible to have various configurations, applications or different calculator formats available in your **RSU**. For example, with one function you could switch between a financial and a technical calculator. In addition, these functions can be very useful should you wish to save the calculator's memory untouched whilst you perform general calculations during the running of an application program.

Extended Memory is not affected by these functions.

## EXALL — EXchange ALL main memory

XROM 04,13

Details: The content of the specified type "W" (Write All) RSU-file and the entire content of the HP-41's main memory are exchanged.
If the Autostart flag (flag 11) was set when the specified type "W" RSU-file was stored with **STOALL**, or by a previous **EXALL**, then program execution will automatically begin at the program and line number set by the program counter at creation time. *See also STOALL.*

Input: The Alpha-register should contain the name of the type "W" RSU-file to be used.
File names are limited to between 1 and 7 characters in length.

Errors: Refer to the *HP-41 Owners Manuals* for details of displayed standard messages.

"CONFIG ERR" The file specified in Alpha is not the same size as the 41C's main memory. Plug in or unplug memory modules as necessary.

"FL TYP ERR" The file specified in Alpha is not a type "W" file. Use the FLDIR function to check that the file exists.

"NAME ERROR" An illegal file name was specified in the Alpha-register. File names of 1 to 7 characters are allowed.

"NO FL NAME" No file name was specified in Alpha.

"NOT A FILE" The file name given in Alpha does not currently exist.

# RCLALL

**ReCaLl ALL main memory**

**XROM 04,26**

**Details:** The entire main memory is replaced by the content of the specified type "W" (Write All) RSU-file.
Automatic execution of programs is possible by setting the Autostart flag (SF 11) when the type "W" RSU-file is created. *See also STOALL.*

**Input:** The Alpha-register should specify the type "W" RSU-file to be used. File names are limited to between 1 and 7 characters.

**Output:** The entire content of main memory is replaced by the type "W" RSU-file.
If flag 11 was set when the "W" file was stored by **STOALL** or **EXALL**, then program execution will automatically begin with the line and program specified by the program pointer at the time of original creation.

**Errors:** Refer to the *HP-41 Owners Manuals* for details of displayed standard messages.

"CONFIG ERR" The file name specified in the Alpha-register is not the same size as the HP-41C's main memory. Plug in or unplug memory modules as necessary.

"FL TYP ERR" The file specified in Alpha is not a type "W" RSU-file. Use FLDIR to check that the named file exists.

"NAME ERROR" An illegal file name was specified in the Alpha-register. File names of 1 to 7 characters are allowed.

"NO FL NAME" No file name was specified in Alpha.

"NOT A FILE" The file named in the Alpha-register does not currently exist.

## STOALL — STOre ALL main memory

**XROM 04,43**

Details: The entire contents of the HP-41's main memory is stored in a special RSU-file of type "W" (Write All).
The length of the file will depend upon the particular HP-41 model and the amount of extra memory plugged in. For the basic **41C**, the size will be 80-registers. Each single memory module will add another 64 registers, up to the maximum of 336 registers. For the **41CV** or **41CX**-models, the file size is 336 registers.
In common with the Card Reader **WALL** function, the Autostart flag (flag 11) may be combined with **STOALL** to automatically begin running a program when the main memory file is later recalled from the RSU. To activate this feature, set flag 11 (**SF11**), position the program counter to the particular program line from which the program is to begin running, and then execute **STOALL**. The position of the program counter can easily be checked by activating PRGM-mode and examining the line number displayed.

Input: The Alpha-register should contain a file name of between 1 and 7 characters in length.

Errors: Refer to the *HP-41 Owners Manuals* for details of displayed standard messages.

"DUP FL NAME" The specified file name already exists. Use FLDIR to check existing names and select a new, unique name.

"NAME ERROR" An illegal file name was specified in the Alpha-register. File names of 1 to 7 characters are allowed.

| | |
|---|---|
| "NO FL NAME" | No file name was specified in the Alpha-register. |
| "NO ROOM" | The remaining free registers in any one RSU-data block is less than the number needed. Free additional data registers by deleting a file, or initialise a new data block. |
| "NO RSU PAGE" | No RSU-block is currently enabled. Check the RSU-switches.<br>RSU-blocks are either not initialised, or have been initialised for program storage, but not for data storage. |

# Section 10

# PROGRAM FUNCTIONS

Program Functions include all functions designed for working with user code programs (RPN) in a **RSU** block and can be subdivided into two groups:

**Transfer Functions** facilitate the transfer of a user code program from main memory to the first empty, unsecured Program Block within the RSU.

| | |
|---|---|
| **CLPR** | CLear Program in Rsu |
| **LOADP** | LOAD Program to program block |

The user may copy programs from the RSU-program block to main memory by using the standard **COPY** function.

**Security Functions** control whether loading and deletion of programs is possible within a specified program block.

| | |
|---|---|
| **CLSEC** | CLear SECured status |
| **SETSEC** | SET prgm block SECured |

## RSU Program Blocks

The **ES-41 DATABASE** module allows the user to partition the ERAMCO SYSTEMS RSU into program blocks that can be used to emulate a plug-in application module.

In designing the HP-41, HP allowed for system extensions by the use of XROM (eXternal ROM) reference numbers. Every plug-in module has an XROM identity (a number from 00 to 31) and individual functions, including RPN programs, within that module have their own Function Address Table (FAT) number up to 63. When an external function or a program name is entered as a line in a program, the actual name is not stored, but rather the corresponding XROM number is. When you edit or run the

program, providing the plug-in module is still inserted, the 41's operating system is able to decode the XROM number and displays the function's name for your benefit. If, however, the module is removed from the 41, then the operating system has no way to decode the XROM number.

When your initialise a program block within the RSU, you are prompted for an XROM identity (e.g. 10). Each program that you subsequently store in that program block is sequentually assigned a number in that block's FAT.

For example, lets assume you have written a program called "FUEL" and used **LOAP** to load it as the 13th program into a RSU-block initialised as XROM 10. Program "FUEL" will then become XROM 10,13. While writing another program, "BANKACC", the "FUEL" program is used as a sub-routine, e.g. XEQ "FUEL". As an external ROM function/program that line will be displayed as XROM "FUEL", but if the RSU is removed, and that line is again examined, it will be seen to really be stored as XROM 10,13.

Because programs within a particular block are sequentually numbered, users should be very careful when deleting programs from within a block. Imagine that we had used **LOADP** to load a total of 15 programs in one block:

| *XROM* | *Program Name* |
|---|---|
| 10,01 | COMET |
| : | |
| : | |
| 10,11 | INCOME |
| 10,12 | TEST1 |
| 10,13 | FUEL |
| 10,14 | ADDRESS |
| 10,15 | BANKACC |

For some reason, we decide that we don't want the "TEST1" program any more, so use the **CLPR** to clear the program from the block. Because the Function Address Table must have sequential numbers, all other programs in the block are moved down in the table:

| *XROM* | *Program Name* |
|---|---|

```
10,01        COMET
 :
 :
10,11        INCOME
10,12        FUEL
10,13        ADDRESS
10,14        BANKACC
```

We now encounter a problem. Our program, "BANKACC", calls the "FUEL" program as a sub-routine, but remember the instruction actually stored in the program is XROM 10,13 and not XEQ "FUEL". In deleting "TEST1", all other programs in the FAT were also renumbered and program "FUEL" is now XROM 10,12, not XROM 10,13. The result is that "BANKACC" will no longer call "FUEL", but will instead call the program "ADDRESS" as this is now XROM 10,13. To correct the problem, we must edit the "BANKACC" program and replace all occurrences of XROM 10,13 by keying XEQ "FUEL" again.

Whilst admittedly some difficulties may be caused, with good programming practice of documenting your programs as they are written, you should experience no real problems. As such, the difficulty is neither associated with the **ES-41 DATABASE** module, nor the RSU, but is rather a limitation of the HP-41's operating system. The same difficulty will be encountered when using plug-in application modules with the same XROM identity.

The user should be particularly aware of this problem when calling other RSU-programs as sub-routines, and when assigning RSU-programs to USER-mode keys.

The total number of FAT entries is 64, including the block header assigned when the block is first initialised using INIPRGM. In addition to the FAT entries used for each program, it should be remembered that an additional entry is made for every global alpha label (excluding local labels, *A* to *J* and *a* to *e*) contained in the program. Extensive use of global alpha labels will quickly exhaust FAT entries.

## CLPR

**CLear PRogram in rsu**

**XROM 04,03**

Details: The named program is cleared from the RSU-program block containing that program. It will remove the first named program found in the program blocks and then pack the block that contained the program to free up as much space as possible. When the program is deleted, the entry for that program in catalogue (**CAT 2**) is also deleted.

Cautions: **When there are two programs of the same name existing in different program blocks, only the program appearing first in CATalogue 2 will be cleared. The user should be aware of the difficulties that can be caused by references, e.g. calling the deleted program, contained in other programs.** *See also pages 133 to 135.*

Input: The user is prompted for the name of the program to be deleted in the same way as **CLP**.

Output: The named program is deleted and all other programs in the same program block are packed. FAT (Function Address Table) entries are also moved, where necessary to reflect the deletion.

Errors: Refer to the *HP-41 Owners Manuals* for details of displayed standard messages.

"NO PRGMROM" The named program resides in a HP-41 plug-in application module (ROM).

"PRGMROM SEC" The RSU-program block containing the named program is secured against deletion. Use **CLSEC** to unsecure it.

"RAM" The named program also exists in HP-41 main memory. Either clear, or rename this RAM program to be able to delete the RSU-program.

# LOADP

**LOAD Program to program block**

**XROM 04,24**

Details: The named program in main memory will be loaded into the first RSU-program block with enough free registers and FAT (Function Address Table) entries to hold the program.
Before loading, the program will be packed to remove unused space (indicated by the message "PACKING"). If the last instruction in the program is not an END, this will be added.
After packing, a "LOADING PRGM" message will be displayed.
If loading is successful, the program will be identified in the FAT by the next sequentually available XROM number. FAT entries will also be made for all global alpha labels (except local labels *A* to *J* and *a* to *e*) in the program.
If room for the program is not available in any of the initialised RSU-program blocks, the function will abort with the message "NO ROOM" and return the maximum number of free registers found in any one program block to the X-register. The most commonest cause is that several large programs have been loaded and the remaining space is not quite large enough for the current program.
If there are inadequate FAT entries remaining in the initialised program blocks for the number of global labels contained in the program, loading will abort with the message "NO PRGMROM". In this case the X-register value is not changed.

Cautions: **Do not load two programs of the same name into program blocks as only the program appearing first in CATalogue 2 can be accessed.**

Input: At the prompt, enter the name of the program to be loaded into the RSU-program block.

Output: If adequate room is not found in any one block, the greatest number of free registers in any program block is returned to the X-register.

Errors: Refer to the *HP-41 Owners Manuals* for details of displayed standard messages.

"PRGMROM SEC" The Program Block containing the named program is secured. It should first be unsecured using **CLSEC**.

"NO ROOM" Inadequate space is available in any one block to hold this program. Delete other programs or create a new program block with **INIPRGM**.

"NO PRGMROM" There is no program block in the system. Initialise using **INIPRGM**.
There are not enough FAT entries remaining in any one block for the current program's global labels. Either delete other programs, initialise a block, delete global labels or change them to local labels.

# CLSEC

**CLear SECured status**

**XROM 04,05**

Details: A flag in the specified program block is reset to unsecured status, thereby allowing programs to be loaded into, or deleted from that block.
It is not neccessary to unsecure a program block in order to be able to re-initalise the secured program block as a data or program block.

Input: The X-register should contain the number of the RSU-program block to be unsecured. A positive value (+1 to +4) specifies a lower port block, or a negative value (-1 to -4) an upper block.

Output: The specified program block is unsecured.

Errors: Refer to the *HP-41 Owners Manuals* for details of displayed standard messages.

"ALPHA DATA" The X-register contains alphanumeric data. Change to a valid port block.

"BAD PAGENO" The port block number specified in the X-register is invalid and does not correspond to an enabled RSU-block.

"NO RSU PAGE" No RSU-blocks are enabled. Check the RSU-switches.
Blocks are either not initialised, or have been initialised as data storage blocks, but not as program blocks.

"NO PRGMROM" The block specified in the X-register is not a program block. Change to a valid program block or use **INIPRGM** to create a program block.

"NONEXISTENT" An illegal number, less than -999 or more than +999, was specified in the X-register.

# SETSEC

**SET prgm block SECured**

**XROM 04,41**

Details: A flag in the specified program block is set to secured status thereby preventing programs from being loaded into, or deleted from that block. Securing blocks may be used when two program blocks are present in the RSU system, and it is desired to force loading of programs to one particular block.

Cautions: **Although secured, the block may still be re-initialised by INIDATA or INIPRGM.**

Input: The X-register should contain the RSU-program block number to be secured. A lower port block may be specified by a positive value (+1 to +4), or an upper block by a negative value (-1 to -4).

Output: The specified program block is secured.

Errors: Refer to the *HP-41 Owners Manuals* for details of displayed standard messages.

"ALPHA DATA" The X-register contains alphanumeric data. Change to a valid port block.

"BAD PAGENO" The port block specified in the X-register is invalid and does not correspond to an enabled RSU-block.

"NO RSU PAGE" RSU-blocks disabled. Check switches. RSU-blocks have been initialised as data blocks, not as program blocks.

"NO PRGM ROM" The block specified in the X-register is not a program block. Change to a valid program block number or use **INIPRGM** to create a program block.

"NONEXISTENT" An illegal number, less than -999 or more than +999, was specified in the X-register.

# Section 11

# PROGRAM EXAMPLES

With a set of functions as rich as those provided by the **ES-41 DATABASE SOFTWARE**, and an almost limitless possibility for their application, it is very difficult to provide complete applications programs that would be of value to more than a small percentage of users. This is particularly true of database applications where, for example, one user may concentrate on storing numeric data of one value per record, whilst another user may store alphanumeric data consisting of multi-field records.

Because of this, and the additional complexity caused by the differing equipment requirements of users, we have not tried to provide complete applications programs. Instead, this section contains a few example programs to illustrate the use of some of the **ES-41 DATABASE** functions.

---

**1.** **This example illustrates storing values from the keyboard into a Data File.**

The program first creates a new data file by prompting for the name and number of items to store in that file. Just press the [R/S]-key to restart after each input. The user is then repeatedly prompted for input of an "ART.NAME ?" (article name), followed by a "QTY ?" (quantity). These values are sequentually stored in the data file and the process repeated until either the file is full, or the user just presses the [R/S]-key in response to the "ART.NAME ?" prompt.

```
01*LBL 'INPUT'
02 'ITEMS?'          Prompt for number of items to be input -
03 ASTO X             this being used to generate the file size.
04 'FILE ?'
05 AON
06 PROMPT            Prompts for name to assign to the new file.
07 AOFF
08 STOP              Stop with "ITEMS?" in display to allow
                      input of number of items to be stored in file.
09 ENTER↑            Duplicate number of items
```

```
10 ST+ X          Doubles items to give number of registers
11 CRDF           Creates named file of that size.
12 X<>Y           Recall number of items as DSE counter.
13*LBL 01
14 ENTER↑         Duplicate items counter
15 AON            Activate Alpha-mode
16 CF 23          Clear Alpha input flag
17 'ART.NAME?'    Prompts for article name (maximum of 6
18 PROMPT         characters - because name is stored in a
19 AOFF           single register)
20 FC? 23         Check for input. If clear, no entry made,
21 GTO 02         thus go to the end routine
22 ASTO X         Alpha-Store article name into Reg:X
23 STOIDF         Store name into the DF & increment pointer.
24 'QTY ?'
25 PROMPT         Prompt for input of quantity
26 STOIDF         Store quantity into DF & increment pointer
27 X<> Z          Return DSE counter to R:X
28 DSE X          Decrement counter = remaining registers
29 GTO 01         If non-zero, loop back
30*LBL 02         End routine
31 END
```

Sometimes it is useful to insert a terminating value into the data file to facilitate searching or to indicate the last register used. The following changes can be made to the above program to store "DONE" as the end marker when input of items is terminated before the file is full.

From Line 30 onwards, replace all instructions with:

```
30 ENTER↑         Duplicates DSE counter
31*LBL 02
32 'DONE'
33 ASTO  X        Store "DONE" message in Reg:X
34 DSE Y          Use DSE counter to determine if terminated
                  early at lines 20/21 by pressing [R/S]-key.
35 STODF          If terminated, store end marker into file.
36 END            Stop with "DONE" in display.
```

**2.** The next routine displays and, if a printer is attached, prints value contained within the specified data file. Only the stack registers are used.

```
01*LBL 'DISP'
02 AON
03 'FILE ?'
04 PROMPT          Input name of data file
05 AOFF
06 RSTDF           Reset working file and pointer to DF:00
07 SZ?DF           Recall file size to use as DSE control word.
08*LBL 01
09 RCLIDF          Recall value and increment pointer
10 VIEW X          Display/Print the value
11 FC? 21          Tests for printing enabled.
12 PSE             Pauses if printing not enabled
13 RDN             Roll stack down to return counter to Reg:X
14 DSE X           Decrement counter
15 GTO 01          Loop back if not at end of data file
16 'DONE'
17 AVIEW           Display "DONE" and end.
18 END
```

Where values are stored in pairs, e.g. Article Name and Quantity, it is useful to display/print them on the same line. To do this, make the following changes:

Replace from Lines 08 to 13 inclusive with the following:

```
08 2 E-5
09 +               Creates DSE counter with a step size of 2
10 'ART.    QTY'   Labels the output
11 AVIEW
12*LBL 01
13 CLA             Clear Alpha-register
14 RCLIDF          Recall value & increment pointer
15 ARCL X          Append value to Alpha-register
16 '├: '           Appends ": " to value in alpha.
17 RCLIDF          Recall value & increment pointer.
18 ARCL X          Append value to Alpha register.
19 AVIEW
20 FC? 21          Tests for enabled printer
21 PSE             Replace with STOP if PSE is too short.
22 X<> Z           returns counter to Reg:X
```

**3.** This routine calculates the mean and standard deviation of a given data file.

At the prompt, "FILE ?", input the file name, and press the [R/S]-key. The calculation will then be performed.

Note: Because unused registers within a data file are not really empty, but contain the value "0.0", it is important that these registers are not mistakenly included in the calculations. The user must therefore either employ an end-marker (to indicate the last register used), or ensure that the file size matches the number of valid values/used registers.

The following routine searches the datafile for an end-marker, in this case the string "DONE". If the end-marker is not found, all registers in the file are considered valid; whether these have been used or not.

SIZE =>006 ΣREG set to R:00
Functions used: RSTDF, SZ?DF, RCLIDF, RPTDF, SPTDF, FINDI.

```
01*LBL 'DFSTATS'
02 'FILE ?'
03 AON
04 PROMPT          Prompts for name of data file
05 AOFF
06 RSTDF           Resets working file & pointer to DF:00
07 SZ?DF           Returns the file's size
08 ΣREG 00         Sets statistics registers R:00 to R:05
09 CLΣ             Clears statistics registers
10 1               Sets direction & step size for FIND searching
11 ENTER↑
12 'DONE'          value of end marker to search for.
13 ASTO X
14 FINDI           Searches for the "end" marker
15 GTO 02          If not found, jump to LBL 02
16 RPTDF           Recalls pointer position = end marker
17 0
18 SPTDF           Resets pointer to DF:00
19 ENTER↑
20*LBL 02
21 X<> Z           Use either size or position of "DONE" as end.
22 1E3             }
23 /               } Sets up the ISG counter based upon either
24 1               } size or end-marker position.
25 +               }
26*LBL 01          Loop for accumulating statistics values.
```

```
27 RCLIDF          Recall value and increment pointer position.
28 Σ+              Accumulate value into statistics registers.
29 RDN
30 ISG Y           Increment counter
31 GTO 01          Loop back if non-zero
32 MEAN            Compute mean
33 'MEAN= '
34 ARCL X
35 AVIEW
36 FC? 21
37 PSE
38 SDEV            Compute Standard Deviation
39 'SDEV= '
40 ARCL X
41 AVIEW
42 END
```

---

**4.** This routine sorts into order all values in the named data file.

Note: Alphanumeric values are considered greater than numeric. The entire file is sorted and no check is made for unused registers containing zero.

```
01*LBL 'DFSORT'
02 'FILE ?'
03 AON
04 PROMPT
05 AOFF
06 RSTDF           Reset working data file & pointer to DF:00
07 ASTO X          Stores file name in Reg:X
08 SZ?DF           Recall file size
09 1               Subtract 1 to give last register in file and use
10 -               as control word for DSE.
11 'SORTING..'
12 ARCL Y
13 AVIEW           Display "SORTING..filename"
14*LBL 01
15 RPTDF           Recall pointer value
16 RCLIDF          Recall value & increment pointer position
17 RCL Z
```

```
18 RDN
19*LBL 02
20 X>DF?          Compare R:X against DF register
21 X<>DF          If R:X > DF:xx, then exchanges values
22 IPTDF          Increments pointer by one
23 DSE Z          Decrement control word in Reg:Z
24 GTO 02         If not zero, loop back
25 RDN            Return old pointer value to Reg:X
26 SPTDF          Reset the pointer value to value in Reg:X
27 R↓             Roll stack up to return smallest value found
28 STODF           to Reg:X and store this in DF
29 IPTDF          Increment pointer by one
30 R↓             Roll up stack to return control word for
                  outer loop back to Reg:X
31 DSE X          Decrement register counter
32 GTO 01         Loop back if not yet zero
33 'DONE'         Otherwise display "done" message and
34 AVIEW          terminate execution.
35 END
```

# Appendix A

# OWNER'S INFORMATION

The ERAMCO SYSTEMS **ES-41 DATABASE System** has been specifically designed for use with the HP-41 Handheld Computer and comprises the ERAMCO SYSTEMS **ES-41 DATABASE Module Software** and the **RSU** (RAM Storage Unit). There are two models available:

* 8K-byte RSU with inbuilt **ES-41 DATABASE** Software; or
* 16K-byte RSU with the **ES-41 DATABASE** Software contained in a separate plug-in module.

The following information applies to maintenance of both models.

## RSU Maintenance

During normal, or extended use, the ERAMCO SYSTEMS RSU (RAM Storage Unit) should not require maintenance, other than periodic replacement of batteries. With the exception of these two batteries contained inside the RSU there are no other user serviceable parts.

The RSU-batteries are used to power the memory circuits only when the RSU is **not** connected to the HP-41. When connected, power is supplied from the HP-41 itself. Battery life is therefore dependent upon conditions of use. With fresh batteries, the unit should be powered for at least 6 months when **not** connected to the HP-41. Under conditions when the RSU is more frequently connected to the HP-41, battery life may vary between 6 months and two years.

However, for reasons of data security, it is recommended that batteries are replaced at approximately **six month** intervals.

To prevent damage to the unit, it is strongly advisable to use "leakproof" alkaline batteries in the RSU. The 1.5 volt batteries used are size "UM-3", also known as "penlight", "AA", "MN1500", or "LR6".

## Operating Precautions

**During use, there are several precautions that you should observe to ensure trouble free operation.**

---

CAUTIONS

Do not place fingers, tools, or other foreign objects into any of the HP-41's Input/Output ports. Such action could result in minor electrical shock hazard and interference with pacemaker devices worn by some persons. Serious damage to port contacts and internal circuitry could also result.

Before installing or removing the **RSU**, the **ES-41 DATABASE** module, or any other HP-41 module, be sure to turn off the HP-41 computer. See also the *HP-41 Owner's Manuals.*

The **RSU** and the **ES-41 DATABASE** module can only be inserted one way into the HP-41. Do not try to force it into a port as this could damage contacts in either or both devices.

Protect the HP-41's ports from dust by keeping a port cap installed in any empty port. Also protect the RSU and module connector from damage and contamination by dust and dirt.

---

# LIMITED 180-DAY WARRANTY

The **ES-41 DATABASE System** has been developed and manufactured to the highest possible standards by ERAMCO SYSTEMS. With the exception of software and Owner's Manual content, both products are warranted by ERAMCO SYSTEMS against defects in materials and workmanship affecting electronic and mechanical performance for a period of 180 days from the date of original purchase. If given as a gift, the warranty is transferred to a new owner for the remainder of that period, provided that proof of purchase date is supplied. During the warranty period, ERAMCO SYSTEMS will replace or, at our option, repair a product that proves to be defective, provided that it is returned, shipping prepaid, together with proof of purchase to ERAMCO SYSTEMS or their official service representatives.

ERAMCO SYSTEMS makes no expressed or implied warranty with regard to the software or program material offered, nor to merchantability or fitness of the material for any particular purpose. Software and Owner's Manual material is made available to the user on an 'as' is basis, with the entire risk as to quality and performance resting with the user. Whilst every effort has been made to eliminate deficiencies, the user (and not ERAMCO SYSTEMS, nor any other party) shall bear the entire cost of all necessary correction and all incidental or consequential damages.

The ES-41 DATABASE System is sold on the basis of specifications as at manufacture. ERAMCO SYSTEMS shall be under no obligation to modify or update the **ES-41 DATABASE System** once manufactured.

## Warranty for Consumer Transactions in the United Kingdom

This warranty shall not affect the statutory rights of a consumer whose rights as Buyer and the obligations of Seller are determined by statute.

## Limitations of Warranty

The ERAMCO SYSTEMS warranty does not, and shall not apply if the **ES-41 DATABASE System** has been damaged by accident, misuse, modification, or service by persons or organisations other than official service representatives. No other expressed or implied warranty is or shall have been given. The repair, or replacement of the system is your exclusive remedy. Under no circumstances shall the liability of ERAMCO SYSTEMS exceed the catalogue price of the product at the time of sale.

## Shipping for Service

In the unlikely event that the **ES-41 DATABASE System** proves to be defective, return the system, postage prepaid, to:

ERAMCO SYSTEMS,
W. van Alcmade str. 54,
1785 LS Den Helder,
The Netherlands.

or to other official service representatives.

When returning the system, be sure to include the following items:

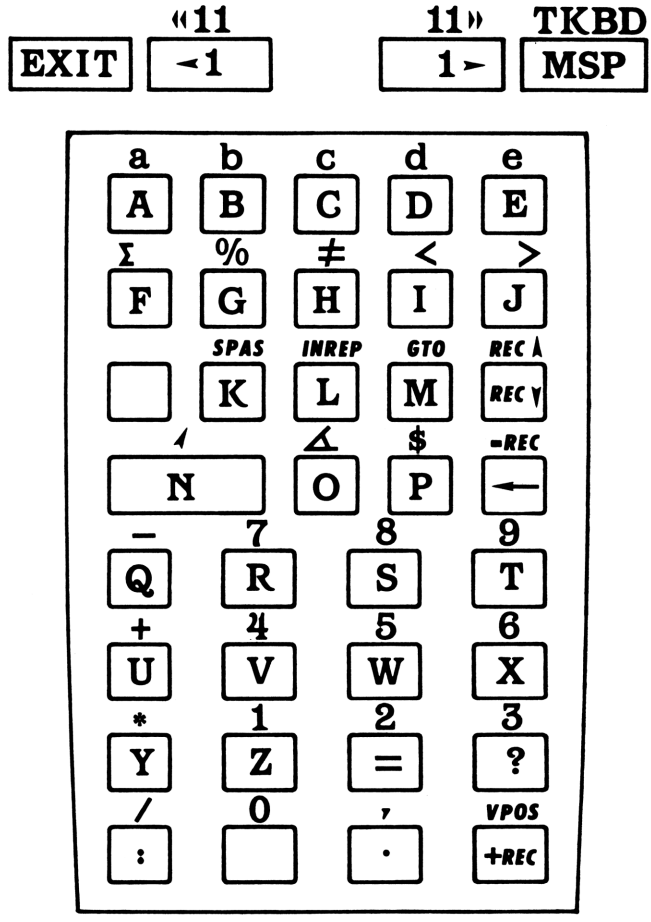
* A Sales Receipt, or other proof of purchase (if the warranty period has not expired).
* A description of the problem, detailing when and how the problem occurs.

Whether or not the **ES-41 DATABASE System** is still under the warranty, it is the responsibility of the owner to ensure that the device is securely packaged to prevent damage in transit (this is not covered by our warranty) and that shipping costs to ERAMCO SYSTEMS are paid.

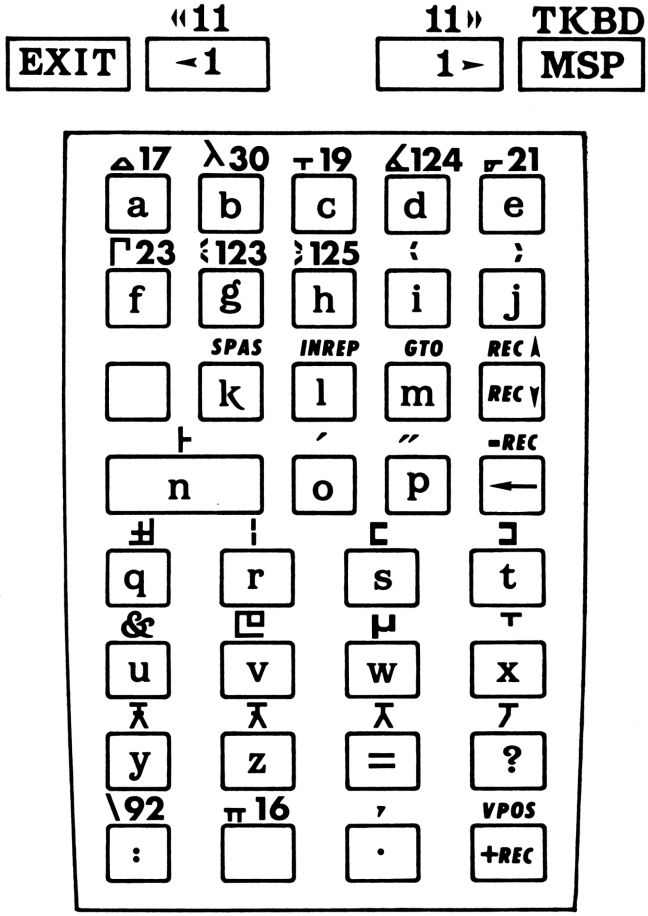
## Technical Assistance

The keystroke procedures, program material and operating instructions provided for using the **ES-41 DATABASE System** are supplied with the assumption that the user has a working knowledge of the concepts, terminology, technology and equipment used. The technical assistance of ERAMCO SYSTEMS, shall be limited to explanation of the operating procedures used in the manual.

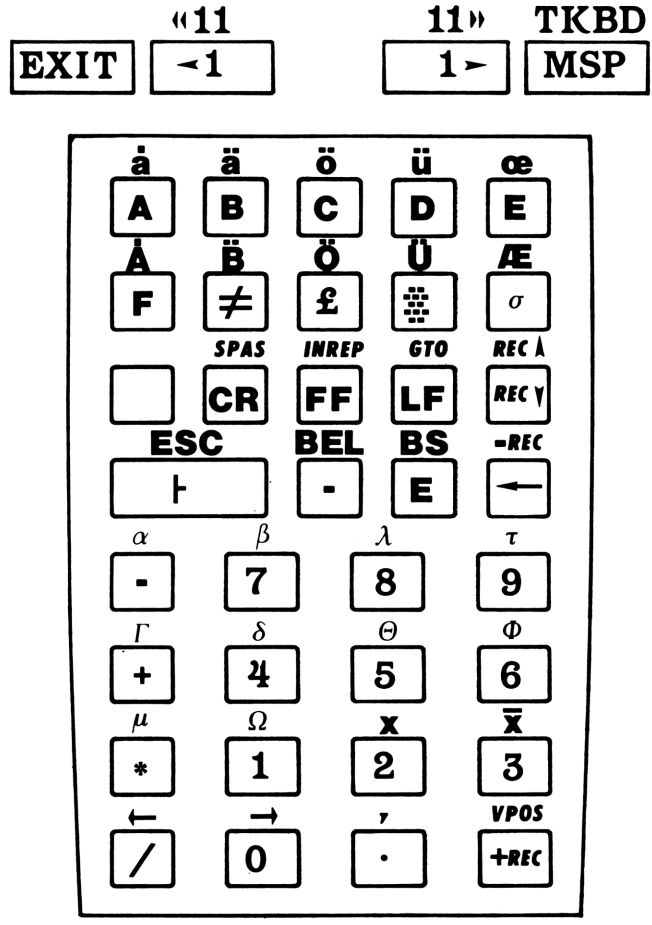
Keyboard 1 for ASCII-file editor : 'Alpha' on



Keyboard 2 for ascii-file editor : 'Alpha' OFF
flag 0 OFF



Keyboard 3 for Ascii-file editor  'Alpha' OFF
flag 0 ON



ACHR Append CHaRacters
XROM 06,01

The characters in Alpha are appended to the end of the current record.

Alpha holds the characters to be appended to the record.

The character pointer is set to the end of the record.

| | |
|---|---|
| FILE FULL | There is not enough room in the working file to insert the characters in Alpha. Use RESAF to lengthen the file or shorten the character string in Alpha. |
| NO RECORD | The record pointer is positioned at record 0. Use IRCPT to set it to record 1. |
| NO WORKFL | There is no current or working ASCII file selected. Use RSTAF to select a working file. |
| REC TOO LONG | The new length of the current record would be too long after inserting the character string in Alpha. Shorten this string, use a new record or shorten the record. |

---

AFLD Append FieLD
XROM 06,02

The characters in Alpha are appended to the end of the current record as a field. The characters are terminated with a delimiter byte. Take in account this byte when adding a field, for there might be just enough room to add the character string, but not the delimiter.

Alpha holds the string of characters to be appended as a field.

The character pointer is set to the end of the record.

| | |
|---|---|
| FILE FULL | There is not enough room in the working file to insert the characters in Alpha. Use RESAF to lengthen the file or shorten the character string in Alpha. |
| NO RECORD | The record pointer is positioned at record 0. Use IRCPT to set it to record 1. |
| NO WORKFL | There is no current or working ASCII file selected. Use RSTAF to select a working file. |
| REC TOO LONG | The new length of the current record would be too long after inserting the character string in Alpha. Shorten this string, use a new record or shorten the record. |

---

ALEN? Alpha LENgth ?
XROM 06,03

The length of the string of characters in Alpha is returned to

the X-register.

None.

The X-register contains the length of the character string in Alpha. If the stack lift was enabled, the stack will be raised and the value in the T-register lost. If the stack lift was disabled, the X-register will be overwritten by the length of the character string in Alpha.

---

AREC Append RECord
XROM 06,04

The characters in Alpha are appended to the end of the working file as a new record. The pointers are set to the end of the record. A record uses one extra byte for indicating the length of the record and the start of it. Therefore it might happen that although there is just enough room to hold the character string, there is not enough room to hold the new record.

Alpha contains the string of characters to be appended as a new file.

The record pointer is set to the new record and the character pointer is set to the end of the record.

| | |
|---|---|
| FILE FULL | There is not enough room in the working file to insert the characters in Alpha. Use RESAF to lengthen the file or shorten the character string in Alpha. |
| NO WORKFL | There is no current or working ASCII file selected. Use RSTAF to select a working file. |

---

ASRM? AScii RooM ?
XROM 06,05

The number of free bytes in the working ASCII file is recalled to the X-register. This number is not neccessarily the number of bytes, that can be used, as the maximum length of a record is 255 characters. Any extra record needs an record length byte and therefore uses extra space.

None.

The X-register contains the number of free bytes in the working file. If the stack lift was enabled, the stack will be raised and the value in the T-register lost. If the stack lift was disabled, the X-register will be overwritten by the number of free bytes in the working file.

| | |
|---|---|
| NO WORKFL | There is no current or working ASCII file selected. Use RSTAF to select a working file. |

---

A=F?  Alpha = Field ?
XROM 06,06

The string of characters in Alpha and the field specified by the X-register are compared. Whenever the X-register is zero the comparison is made on a string of characters at the current character pointer, that has the same length as the string in Alpha.

The X-register specifies the field to be compared. If zero a string of characters with the same length as the string in Alpha is compared starting at the character pointer.
Alpha holds the string of characters to be compared.

If the specified string in the record and Alpha are equal the result of the function will be true, else it is false.
By manual execution
YES The result of the test is true.
NO The result of the test is false
By execution under program control {{PICTURE}}

| | |
|---|---|
| NO FIELD | There is no such field in the record. Specifie an existing field in the X-register. |
| NONEXISTENT | An illegal number, less than -999 or more than +999, was specified in the X-register. |
| NO RECORD | The record pointer is positioned at record 0. Use IRCPT to set it to record 1. |
| NO WORKFL | There is no current or working ASCII file selected. Use RSTAF to select a working file. |

---

A<>F?  Alpha <> Field ?
XROM 06,07

The string of characters in Alpha and the field specified by the X-register are compared. Whenever the X-register is zero the comparison is made on a string of characters at the current character pointer, that has the same length as the string in Alpha.

The X-register specifies the field to be compared. If zero a string of characters with the same length as the string in Alpha is compared starting at the character pointer.
Alpha holds the string of characters to be compared.

If the specified string in the record and the string in Alpha are unequal the result of the function is true, else the result is false.
By manual execution
YES The result of the test is true.
NO The result of the test is false
By execution under program control {{PICTURE}}

NO FIELD  There is no such field in the record. Specifie an

| | |
|---|---|
| | existing field in the X-register. |
| NONEXISTENT | An illegal number, less than -999 or more than +999, was specified in the X-register. |
| NO RECORD | The record pointer is positioned at record 0. Use IRCPT to set it to record 1. |
| NO WORKFL | There is no current or working ASCII file selected. Use RSTAF to select a working file. |

---

A<F? Alpha < Field ?
XROM 06,08

The string of characters in Alpha and the field specified by the X-register are compared. Whenever the X-register is zero the comparison is made on a string of characters at the current character pointer, that has the same length as the string in Alpha.

The X-register specifies the field to be compared. If zero a string of characters with the same length as the string in Alpha is compared starting at the character pointer.
Alpha holds the string of characters to be compared.

If the string in Alpha is alphabetically less then the string specified in the X-register the result of the function is true, else it is false.
By manual execution
YES The result of the test is true.
NO The result of the test is false
By execution under program control {{PICTURE}}

| | |
|---|---|
| NO FIELD | There is no such field in the record. Specifie an existing field in the X-register. |
| NONEXISTENT | An illegal number, less than -999 or more than +999, was specified in the X-register. |
| NO RECORD | The record pointer is positioned at record 0. Use IRCPT to set it to record 1. |
| NO WORKFL | There is no current or working ASCII file selected. Use RSTAF to select a working file. |

---

A<=F? Alpha <= Field ?
XROM 06,09

The string of characters in Alpha and the field specified by the X-register are compared. Whenever the X-register is zero the comparison is made on a string of characters at the current character pointer, that has the same length as the string in Alpha.

The X-register specifies the field to be compared. If zero a string of characters with the same length as the string in Alpha is compared starting at the character pointer.
Alpha holds the string of characters to be compared.

If the string in Alpha is alphabetically less then or equal to the string of characters specified in the X-register the result of the function is true, else it is false.
By manual execution
YES The result of the test is true.
NO The result of the test is false
By execution under program control {{PICTURE}}

| | |
|---|---|
| NO FIELD | There is no such field in the record. Specifie an existing field in the X-register. |
| NONEXISTENT | An illegal number, less than -999 or more than +999, was specified in the X-register. |
| NO RECORD | The record pointer is positioned at record 0. Use IRCPT to set it to record 1. |
| NO WORKFL | There is no current or working ASCII file selected. Use RSTAF to select a working file. |

---

A>F? Alpha > Field ?
XROM 06,10

The string of characters in Alpha and the field specified by the X-register are compared. Whenever the X-register is zero the comparison is made on a string of characters at the current character pointer, that has the same length as the string in Alpha.

The X-register specifies the field to be compared. If zero a string of characters with the same length as the string in Alpha is compared starting at the character pointer.
Alpha holds the string of characters to be compared.

If the string in Alpha is greater then the string specified in the X-register the result of the function is true, else it is false.
By manual execution
YES The result of the test is true.
NO The result of the test is false
By execution under program control {{PICTURE}}

| | |
|---|---|
| NO FIELD | There is no such field in the record. Specifie an existing field in the X-register. |
| NONEXISTENT | An illegal number, less than -999 or more than +999, was specified in the X-register. |
| NO RECORD | The record pointer is positioned at record 0. Use IRCPT to set it to record 1. |
| NO WORKFL | There is no current or working ASCII file selected. Use RSTAF to select a working file. |

---

A>=F? Alpha >= Field ?
XROM 06,11

The string of characters in Alpha and the field specified by the X-register are compared. Whenever the X-register is zero the



comparison is made on a string of characters at the current character pointer, that has the same length as the string in Alpha.

The X-register specifies the field to be compared. If zero a string of characters with the same length as the string in Alpha is compared starting at the character pointer.
Alpha holds the string of characters to be compared.

If the string in Alpha is alphabetically greater then or equal to the string specified in the X-register the result of the function will be true, else it is false.
By manual execution
YES The result of the test is true.
NO The result of the test is false
By execution under program control {{PICTURE}}

| | |
|---|---|
| NO FIELD | There is no such field in the record. Specifie an existing field in the X-register. |
| NONEXISTENT | An illegal number, less than -999 or more than +999, was specified in the X-register. |
| NO RECORD | The record pointer is positioned at record 0. Use IRCPT to set it to record 1. |
| NO WORKFL | There is no current or working ASCII file selected. Use RSTAF to select a working file. |

---

CLAF CLear Ascii File
XROM 06,12

The contents of the entire file are deleted.

Alpha holds the name of the file to be cleared.

The record pointer is set to record 0 and the character pointer is not specified. In fact the file gets the same state as when it is created. The cleared file becomes the working file.

| | |
|---|---|
| FL TYPE ERR | The file specified in Alpha is not an ASCII type file. Use the FLDIR function to check that the file exists. |
| NAME ERROR | An illegal file name was specified in Alpha. File names of 1 to 7 characters are allowed. |
| NO FL NAME | No file name was specified in Alpha. |
| NOT A FILE | The name specified in Alpha is not that of an existing ASCII file. |

---

CRAF CReate Ascii File
XROM 06,13

Creates an ASCII file of the name specified in Alpha and a length specified in the X-register. The created ASCII file becomes the working ASCII file.
The created ASCII file will be placed in the first data-block



with enough free register space available.
The created ASCII file is completely empty. There are no records in the file yet and the record pointer is set to 0. This will cause most functions to display an error. The character pointer is not defined at this stage.

Warnings
An unique file name should be specified.

Alpha should contain a name for the file to be created of between 1 and 7 characters in length. Register-X should contain the length of the ASCII file as a number of registers. The maximum file length that can be created in an empty data block is 679 registers.

An ASCII file of the specified length and name is created. The record pointer is set to 0 and the character pointer is undefined.
If the function causes the NO ROOM error, because not enough free registers are available in any one Data Block, then the X-register contents will be overwritten by the maximum number of free registers available.

| | |
|---|---|
| DATA ERROR | A value of "0" registers was specified as the length of the ASCII file. |
| DUP FL NAME | The specified ASCII file name currently exists. Use FLDIR to check existing names and select a new unique name. |
| NAME ERROR | An illegal file name was specified in Alpha. File names of 1 to 7 characters are allowed. |
| NONEXISTENT | An illegal number, less than -999 or more than +999, was specified in the X-register. |
| NO RSU PAGE | No RSU-block is currently enabled. Check the RSU switches. RSU blocks are either not initialised, or have been initialised as Program Blocks, but not as Data Storage Blocks. |
| NO FL NAME | No file name was specified in Alpha. |
| NO ROOM | The specified length for the file is too long for the amount of free registers remaining in the RSU-Data Block. Either free additional data registers by deleting a file (using DELFL), or resize another file(s). If only one data block is available, the RLEFT? can be used to check the number of free regsters remaining. |

---

DCHPT Decrement CHaracter PoinTer
XROM 06,14

The character pointer is decremented by one. If the character pointer is already at the first character of the record, nothing happens.

None.



The character pointer is decremented by one.

| | |
|---|---|
| NO RECORD | The record pointer is positioned at record 0. Use IRCPT to set it to record 1. |
| NO WORKFL | There is no current or working ASCII file selected. Use RSTAF to select a working file. |

---

DCHR Delete CHaRacters
XROM 06,15

Starting at the character pointer in the current record the specified number of characters is deleted. The character pointer is set before the first character after the deleted ones.
If the number of characters specified in the X-register is greater then the number of characters that are left after the character pointer all characters upto the end of the record are deleted. The character pointer is set to the end of the record.

The X-register specifies the number of characters to be deleted.

None.

| | |
|---|---|
| DATA ERROR | The X-register contains "0" as input. This should be at least 1. Change it and try again. |
| NONEXISTENT | An illegal number, less than -999 or more than +999, was specified in the X-register. |
| NO RECORD | The record pointer is positioned at record 0. Use IRCPT to set it to record 1. |
| NO WORKFL | There is no current or working ASCII file selected. Use RSTAF to select a working file. |

---

DFLD Delete FieLD
XROM 06,16

The field specified in the X-register is deleted.

The X-register specifies the field to be deleted.

The character pointer is set to the first character of the next field. If the deleted field is the last the character pointer is set to the end of the record.

| | |
|---|---|
| DATA ERROR | The X-register contains "0" as input. This should be at least 1. Change it and try again. |
| NO FIELD | There is no such field in the record. Specifie an existing field in the X-register. |

| | |
|---|---|
| NONEXISTENT | An illegal number, less than -999 or more than +999, was specified in the X-register. |
| NO RECORD | The record pointer is positioned at record 0. Use IRCPT to set it to record 1. |
| NO WORKFL | There is no current or working ASCII file selected. Use RSTAF to select a working file. |

---



DRCPT Decrement ReCord PoinTer
XROM 06,17

The character pointer is not changed, unless the length of the newly selected record is shorter then the character pointer is set to. In this case the character pointer is set to the end of the record.

None.

The record pointer is decremented by one.

| | |
|---|---|
| NO RECORD | The record pointer is positioned at record 0. Use IRCPT to set it to record 1. |
| NO WORKFL | There is no current or working ASCII file selected. Use RSTAF to select a working file. |

---

DREC Delete RECord
XROM 06,18

The current record is deleted. The record pointer is not changed. The character pointer is set to the first character of the new record.
If the deleted record is the last record the record pointer is set to the previous record. If the deleted record is the first and only record in the file the record pointer is set to 0 and the character pointer is undefined.

None.

The record pointer remains the same. The character pointer is set to the first character.

| | |
|---|---|
| NO RECORD | The record pointer is positioned at record 0. Use IRCPT to set it to record 1. |
| NO WORKFL | There is no current or working ASCII file selected. Use RSTAF to select a working file. |

---

EDT EDiTor
XROM 06,19

This function activates the ASCII file text editor. It is aspecially handy when a lot of data has to be entered into an

ASCII file. This is manely due to the fact that there are three special keyboards available.
First there is the normal keyboard with all standard HP-41 characters that also can be entered into Alpha. The second keyboard has all lowercase ASCII characters on it and a lott of special HP-41 characters. The third and last keyboard is aspecially meant when a lot of numeric data has to be entered and all kinds of printer control characters.
With the special function keys on the keyboard it is very easy to step through a file, delete or insert characters or records and



to do all other kinds of editing work. These special function keys are active on all three keyboards.
As this function is fully programmable it is very usefull as an input function for ASCII files. If there is a lot of numeric data to be entered that is accompanied by alpha data to, it might be easier to use EDT instead of writing a program that asks for each item seperate and adding it to the file.

When activated the HP-41 display serves as a window in the ASCII file. It displays a portion of 11 charcaters at a time. The portion displayed is controlled by the values of the record and character pointers. The combination of these two pointers gives the position of the cursor. The record pointer might be seen as a number that controls the vertical position (line) of the cursor, whereas the character pointer controls the position in the horizontal direction (column).
The cursor will be displayed at the position determined by the record and character pointer at entry. This position can be set before activating the editor mode, or can be the same position as when edit mode is left the last time.
When the editor mode is exit the record and character pointers will reflect the position of the cursor.
The cursor is always displayed before a character. The only exception to this rule is when the cursor is at the end of a record. In this case it is positioned after the last character in a record.

If possible the cursor is positioned at the middle of the display. Moving the cursor to the right one place will cause the display to be updated and as the cursor is set in the middle it appears as if the text is moved to the left. This also applies vice versa.
In case the cursor is positioned at the start of the record this will be different. In this case the record number is shown at the left hand side of the display with a small "T" to the right of this number. The cursor is positioned directly to the right of this small "T". Entering characters causes the cursor to move to the right, until it is in the middle of the display again. Entering more characters cause the just entered text to be pushed to the left, as is the normal case.

When the text editor is activated with an empty file or if the record pointer is positioned at 'record 0' the name of the file

that is edited is displayed.

The HP-41 display is also used to display information about the modes the text editor is operating in. These modes are controlled by the special function keys, but can be controlled partly by the setting of some user flags.
To display the different modes there are three annunciators of the display use. The used annunciators are the "0", "1" and the "ALPHA" annunciators. When these annunciators are vissible the described mode is active.

"0" This annunciator displays which of the special keyboards



is active. When lit the numeric/printer keyboard is activated when the editor is set to the special keyboards.
If this annunciator is off the special keyboard that will be activated when switched to the special keyboards is the lowercase keyboard.

"1" This annunciator indicates whethet the editor is in insert or replace mode. When this annunciator is lit the editor is in insert mode. In this case all entered characters will be inserted in front off the character where the cursor is positioned at. The following characters are pushed towards the end of the record.
When this annunciator is off the text editor is in replace mode. In this case the characters that are entered will overwrite the characters the cursor is positioned at.

"ALPHA" This annunciator indicates if one of the special keyboards is activated or if the normal keyboard is active. When this annunciator is on the normal alpha keyboard is active. If the annunciator is off one of the two special keyboards is active. The "0" annunciator is used to display which of the special keyboards is active.

All the other annunciators remain the same as they where when the editor was activated. Therefore the "PRGM" annunciator will lit when the editor is execute from a program, "USER" will be lit if the user mode is on etc.

During the edit mode there is made use of a special insert buffer to speed up editing as much as possible. This means that in normal cases the text can be entered as quick as possible. Only when the cursor is moved the actual file is updated and the insert block is placed in the file. In this way a minimum delay is quaranteed during editing.
When deleting characters with the <-- key the following text is not moved back. This will be done as soon as the cursor is moved. If new text is entered at this place the empty space is used first and when completely filled the insert buffer is used.

First we will describe the special function keys that are common to all three keyboards. These special function keys can be seperated into three different groups.
The first group contains the cursor control keys. These keys are all autorepeat.
The second group in the special function keys contains block functions to operate on records as a whole.
The last group contains some miscellaneous functions. These functions are controlling the operation modes of the editor and some other stuff.

## CURSOR CONTROL KEYS

11

< 1 — With this key the cursor is set one character to the left. If the cursor is already at the first character in the record nothing will happen.

<< 11 — This key serves the same purpose as the non shifted USER key. However instead of stepping one character to the left it steps 11 characters at a time to the left. If the cursor is already at the first character of the record nothing happens.

1 > — Pressing this key advances the cursor one place to the right. If the cursor is already at the end of the record nothing happens.

11 >> — This key advances the cursor 11 places to the right. If the cursor is within 11 characters from the end of the record the cursor is positioned after the last character in the record.

<-- — This is not really a cursor key, but it deletes characters to the left of the cursor. Also the cursor is updated and moved if neccessary.
If there is an error message in the display pressing this key once removes the message from the display and resumes normal operating mode again.

The above mentioned keys control the movement within a record. They only affect the character pointer. If the cursor is moved to the right, the character pointer is incremented. If the cursor is moved to the left the character pointer is decremented.
Holding the keys pressed for over half a second will cause them to go into autorepeat mode. The cursor is stepped every .2 seconds one or more places to the left or right.

REC ^ — This key advances the cursor one record. When the cursor is already at the last record of the file nothing happens.
The character pointer is not changed, unless the value of the character pointer is such that it would be over

the end of the new record. In this case the character pointer is set to the end of the new record.

REC ! This key sets the cursor to the previous record. When the record pointer is set to record 0 the file name is displayed. If the cursor is already at record 0 nothing happens.
The character pointer is not changed, unless the value of the character pointer is such that it would be over the end of the new record. In this case the character pointer is set to the end of the new record.
Whenever the cursor is set to record 0 the character pointer is set to the first character.

These keys normally affect the record pointer only. However, if the character pointer would be positioned over the end of the



record when the cursor is moved up or down, the character pointer is set to the end of the new record.
Holding the keys pressed for over half a second will cause them to go into autorepeat mode. The cursor is stepped every .2 seconds one or more places to the left or right.

BLOCK KEYS

-REC The current record is deleted. The cursor is set to the next record at the first character. If the deleted record is the last record in the file the cursor is positioned at the first character of the previous record.
When the deleted record is the only record in the file the cursor is set at record 0 and the name of the edited file is displayed.
Files are immediately packed when a record is deleted to leave as much space as possible for other text.

+REC With this key a record can be added to the file. The record is inserted after the current record. The cursor is set to the end of the newly created record.
If there is not enough room in the file left to hold a new record the message "FILE FULL" is displayed. Pressing <-- once deletes the error message. There should be made space by deleting records or resize the ASCII file with RESAF.

SPAS This key serves a double purpose. It can be used to split a record into two different records or to concatenate two records to one record. Which function is performed depends on the current cursor position.
When the cursor is somewhere in the middle of a record pressing this key causes the record to be split into two different parts. The part after the cursor will be placed into a new record that comes after the record the cursor is positioned at. The cursor position

remains unchanged.
The error "FILE FULL" might occur when this function is executed. It only happens when the file is completely filled up to the last byte. As the new record needs one extra byte to indictae the start of the record there is no room to accept the new record. Resize the file or delete something in the file.
If on the contrary the cursor is at the end of a record pressing this key will concatenate this and the next record to one record. The position of the cursor remains unchanged.
Pressing this key when the cursor is at the last record of the file causes the error message "NO NEXT REC".

MISCELLANEOUS KEYS

GTO This key allows you to move the cursor to any position in the file without having to step to it with the



cursor control keys.
After pressing this key a three digit prompt appears. Now there are two things that can be done. First a three digit number can be entered. This will position the cursor at the first character of the specified record. If the record does not exist the record pointer is set to the last record in the file.
Pressing the decimal point first will position the cursor at the specified character within a record. Entering zero at this stage causes the cursor to be set after the last character in the record, as entering a number greater then there are characters does.

VPOS Pressing this key displays the current cursor position in terms of a record number and character number within the record. Pressing the <-- key once restore the normal operating mode.

INREP This key toggles the insert replace mode of the text editor. The "1" reflects the current mode. If lit the editor is in insert mode. Pressing the key once will activate the other mode, pressing it another time restores it.
The active mode can be set before entering the editor. This is done by setting or clearing user flag 1 as wished.

MSP Pressing this key toggles between the normal and one of the two special keyboards. The "ALPHA" annunciator reflects which keyboard is active. If lit the normal keyboard is active, else one of the two special keyboards is active.

TKBD This key toggles the two special keyboards. The "0" annunciator reflects which special keyboard is active

when ALPHA is pressed. If "0" is on the numeric/printer keyboard will be activated, else the lowercase keyboard.

EXIT The last special function key is used to exit the editor mode. The calculator is not switched off. The character and record pointer are updated according to the current cursor position. User flag 0 and 1 are also updated to reflect the state of the special keyboards and the insert/replace mode.
The text in the insert buffer is placed into the file and the file is packed to leave as much space as possible.

Following are the three keyboard overlays of the text editor. Each keyboard has its own figure. Characters that can be displayed on the HP-41 are shown as characters. Some of the special characters do have an ASCII code with them.
Many characters are included on the keyboard that are displayed as boxed stars by the HP-41. In this case only the ASCII code is



shown. There is one exception to this. The lowercase characters are shown as normal characters and not as ASCII codes.
With some newer versions of the HP-41 calculator the lowercase characters are also displayed correctly. These newer versions can be recognized in two ways.
First the display has a little mask on it with round corners instead of the normal square corners. The second and more versatile way is to look at the state of user flag 20. The ES-41 Database module will set this flag automatically when it is used with one of the newer HP-41 models.
As on the HP-41 itself the non shifted characters are shown on the keys themself, as the shifted characters are displayed above the corresponding keys.
On the two special keyboards the decimal point and comma depend on the setting of flag 28. If flag 28 is clear, the non-shifted characters will be the comma. If on the other hand user flag 28 is set, the non-shifted character will be the period. The shifted characters are the period and comma respectively.

NORMAL KEYBOARD
"ALPHA" on

LOWERCASE KEYBOARD
"ALPHA" off, "0" off

NUMERIC/PRINTER KEYBOARD
"ALPHA" off, "0" on

Input
If there is a working file active, this working file will be used to edit. When there is no working file active the file to be edited can be specified in Alpha. This file will also become the working file.

With user flags 0 and 1 the initial mode of the editor can be set. With flag 1 the insert replace mode is controlled and with flag 0 the selected special keyboard is controlled. See also the function keys INREP and TKBD.

Output
User flag 0 indicates which special keyboard is selected and user flag 1 indicates whether insert or replace mode is selected. The record and character pointer are set to the last used cursor position.

| | |
|---|---|
| FL TYPE ERR | The file specified in Alpha is not an ASCII type file. Use the FLDIR function to check that the file exists. |
| NAME ERROR | An illegal file name was specified in Alpha. File names of 1 to 7 characters are allowed. |
| NO FL NAME | No file name was specified in Alpha. |
| NOT A FILE | The name specified in Alpha is not that of an existing ASCII file. |

---

FINDPS FIND PoSition



XROM 06,20

This function enables you to search the working file for a match of the string in Alpha. If a match is found the record and character pointers are set to the first occurence of that string in the working file.
The search is controlled by the X-register and the Y-register.
The number in the X-register serves two purposes. The actual number (without sign) controls where the search is performed. The sign of the X-register controls error behavior.
If the number in the X-register is a non zero number, every specified record will be searched starting at the first character of the field specified in the X-register. Alpha is compared with the field as a whole. If a match is found the search is stopped and the pointers are set to the field.
When the value in the X-register is zero a search is performed on all specified records, starting at the current character position. Alpha is compared with the record starting at the current character pointer. If a match is found the pointers are set to the record holding the searched string.
The sign of the X-register controls the error behavior of the function. As not all records are of the same length, or do have the same number of fields it coul happen that a comparison is made within a record, that has less characters or fields then specified. In this case an error situation occurs.
When the X-register is negative, no error occurs and the record is treated as if there is no match with the string to be compared. Whenever the number in the X-register is positive the search is stopped when a record is encounterd that is shorter then specified , or has less fields. In this case the value of the record pointer where the error occured is returned to the X-

register. The pointers are not moved.
The Y-register specifies the records to be searched. Records can be searched in following order or just one record for every number of specified records. The number in the Y-register specifies the step size of the search.
The sign of the Y-register controls the search direction. A negative value causes a backwards search through the file and a positive value causes a forward search.

The X-register specifies the field to compare in each record. If zero the current character position is used. The sign of the X-register controls error behavior: If negative no error occurs when a record does not have the specified field or has less characters then indicated by the current character pointer. If positive the function aborts when encountering such a record and returns the error record number to the X-register.
The Y-register controls the stepsize. If the number is negative the search is backwards, else it is forwards.

The character and record pointer are set to the searched string if it is found, else they are not changed.
When the searched string is found the next step in the program is executed and when executed from the keyboard the message YES is displayed. If the searched string is not found the next program



step is skipped and when executed from the keyboard the message NO is displayed.

| | |
|---|---|
| DATA ERROR | The Y-register contains "0" as input. This should be at least 1. Change it and try again. |
| NO FIELD | There is no such field in the record. Specifie an existing field in the X-register. |
| NONEXISTENT | An illegal number, less than -999 or more than +999, was specified in the X or Y-register. |
| NO RECORD | The record pointer is positioned at record 0. Use IRCPT to set it to record 1. |
| NO WORKFL | There is no current or working ASCII file selected. Use RSTAF to select a working file. |

---

FLEN? Field LENgth ?
XROM 06,21

The length of the field specified in the X-register in the record where the record pointer is positioned at is recalled to the X-register.

In the X-register the field is specified of which you want to know the length.

The X-register contains the length of the specified field in the current record. If the stack lift was enabled, the stack will be raised and the value in the T-register lost. If the stack lift was disabled, the X-register will be overwritten by the number of

fields in the current record.

| | |
|---|---|
| DATA ERROR | The X-register contains "0" as input. This should be at least 1. Change it and try again. |
| NO FIELD | There is no such field in the record. Specifie an existing field in the X-register. |
| NONEXISTENT | An illegal number, less than -999 or more than +999, was specified in the X-register. |
| NO RECORD | The record pointer is positioned at record 0. Use IRCPT to set it to record 1. |
| NO WORKFL | There is no current or working ASCII file selected. Use RSTAF to select a working file. |

---

GCHR Get CHaRacters
XROM 06,22

This function clears Alpha and transfers the number of characters specified in the X-register to Alpha. Transfer is started ast the current character position and is ended when the end of the record is reached or the specified number of characters has been transferred.

The X-register contains the number of characetrs that has to be transferred.



User flag 17 is cleared when the end of the record is reached during transfer of the specified number of characters, else it is set.
The character pointer is set after the last transferred character or when the end of the record is reached during transfer to the first character of the next record. The record pointer is also set to this record.

| | |
|---|---|
| DATA ERROR | The X-register contains "0" as input. This should be at least 1. Change it and try again. |
| NONEXISTENT | An illegal number, less than -999 or more than +999, was specified in the X-register. |
| NO RECORD | The record pointer is positioned at record 0. Use IRCPT to set it to record 1. |
| NO WORKFL | There is no current or working ASCII file selected. Use RSTAF to select a working file. |

---

GFLD Get FieLD
XROM 06,23

This function clears Alpha and transfers the field specified in the X-register to Alpha. Transfer is started at the first character of the field specified in the X-register.

The field to be transferred to Alpha is specified in the X-register.

When the end of the field is reached during transfer user flag 17 is cleared, else it is set.
The character pointer is set after the last transferred character of the field. If the end of the record is encountered during transfer, the record pointer is advanced by one and the character pointer is set to the first character of this record.

| | |
|---|---|
| DATA ERROR | The X-register contains "0" as input. This should be at least 1. Change it and try again. |
| NO FIELD | There is no such field in the record. Specifie an existing field in the X-register. |
| NONEXISTENT | An illegal number, less than -999 or more than +999, was specified in the X-register. |
| NO RECORD | The record pointer is positioned at record 0. Use IRCPT to set it to record 1. |
| NO WORKFL | There is no current or working ASCII file selected. Use RSTAF to select a working file. |

---

GREC Get RECord
XROM 06,24

This function clears Alpha and transfers the characters in the current record to Alpha. Transfer is started at the current character pointer and ended after either the end of the record is reached or Alpha is filled, e.g. 24 characters are transferred to Alpha.



None.

The character pointer is set after the last transferred character.
If the end of the record is reached before Alpha is filled, user flag 17 is cleared, else it is set.
When the end of the record is reached the record pointer is advanced to the next record and the character pointer is set to the first character of that record.

| | |
|---|---|
| NO RECORD | The record pointer is positioned at record 0. Use IRCPT to set it to record 1. |
| NO WORKFL | There is no current or working ASCII file selected. Use RSTAF to select a working file. |

---

ICHPT Increment CHaracter PoinTer
XROM 06,25

The character pointer is advanced by one.

None.

The character pointer is set to the next character.

END OF REC The character pointer would be positioned over the end of the record.
NO RECORD The record pointer is positioned at record 0. Use IRCPT to set it to record 1.
NO WORKFL There is no current or working ASCII file selected. Use RSTAF to select a working file.

---

ICHR Insert CHaRacters
XROM 06,26

The character string in Alpha is inserted to the left of the current character pointer.

Alpha contains the characters to be inserted into the current record at the current character pointer position.

The character pointer is set after the last inserted character.

FILE FULL There is not enough room in the working file to insert the characters in Alpha. Use RESAF to lengthen the file or shorten the character string in Alpha.
NO RECORD The record pointer is positioned at record 0. Use IRCPT to set it to record 1.
NO WORKFL There is no current or working ASCII file selected. Use RSTAF to select a working file.
REC TOO LONG The new length of the current record would be too long after inserting the character string in



Alpha. Shorten this string, use a new record or shorten the record.

---

IFLD Insert FieLD
XROM 06,27

The characters in Alpha are inserted in the current record as a new field to the left of the field that is specified in the X-register.

Alpha contains the characters to be inserted in the current record as a new field.
The field is inserted to the left of the field specified in the X-register.

The character pointer is set after the delimiter byte of the inserted field seperating the inserted field and the field specified in the X-register.

DATA ERROR The X-register contains "0" as input. This should be at least 1. Change it and try again.
FILE FULL There is not enough room in the working file to

| | |
|---|---|
| | insert the characters in Alpha. Use RESAF to lengthen the file or shorten the character string in Alpha. |
| NO FIELD | There is no such field in the record. Specifie an existing field in the X-register. |
| NONEXISTENT | An illegal number, less than -999 or more than +999, was specified in the X-register. |
| NO RECORD | The record pointer is positioned at record 0. Use IRCPT to set it to record 1. |
| NO WORKFL | There is no current or working ASCII file selected. Use RSTAF to select a working file. |
| REC TOO LONG | The new length of the current record would be too long after inserting the new field. Shorten the field, use a new record or shorten the record. |

---

IRCPT Increment ReCord PoinTer
XROM 06,28

The record pointer is advanced to the next record.
The character pointer is not moved, unless the new record has less characters than the character pointer is indicating. In this case the character pointer is set to the last character of the newly selected record.

None.

The record pointer is increased by one. Eventually the character pointer is updated in case the newly selected record is shorter as the character pointer is indicating.

| | |
|---|---|
| NO RECORD | The record pointer is positioned at record 0. Use |



| | |
|---|---|
| | IRCPT to set it to record 1.<br>The newly selected record does not exist. |
| NO WORKFL | There is no current or working ASCII file selected. Use RSTAF to select a working file. |

---

IREC Insert RECord
XROM 06,29

The contents of Alpha is inserted in the working file as a new record. The record will be inserted in front of the current record, allowing to insert a record before the first record. All following records in the file are pushed down to make space for the new record.
The record pointer is set to the newly inserted record and the character pointer is set to the last character in the record.
When Alpha is empty only a new record will be created, with no characters in the record at all.

Alpha contains the characters to be inserted as a new record.

The characters in Alpha are inserted as a new record and the record and character pointer are set to the last character of the just inserted record.

| | |
|---|---|
| FILE FULL | There is not enough room in the working file to insert the characters in Alpha. Use RESAF to lengthen the file or shorten the character string in Alpha. |
| NO RECORD | The record pointer is positioned at record 0. Use IRCPT to set it to record 1. |
| NO WORKFL | There is no current or working ASCII file selected. Use RSTAF to select a working file. |

---

MCHPT Move CHaracter PoinTer
XROM 06,30

The character pointer is moved by the number of characters specified in the X-register. The sign of the X-register determines whether the character pointer is increased or decreased. A negative value decreases the character pointer by the specified number of characters and a positive value increases the character pointer.
The record pointer is not changed. When an attempt is made to move the character before the first caharacter the caharcater pointer will be positioned to the first character of the record.

The X-register contains the number of characters the character pointer has to be moved. If the X-register is negative the character pointer is decreased, else it is increased.

The character pointer is moved by the specified amount of characters in the direction indicated by the sign of the X-register. If the character pointer would be set before the first



character of the record, it is set to the first character of that record.

| | |
|---|---|
| DATA ERROR | The X-register contains "0" as input. This should be at least 1. Change it and try again. |
| END OF REC | The character pointer would be positioned over the end of the record. Change the number in the X-register to a smaller one. |
| NONEXISTENT | An illegal number, less than -999 or more than +999, was specified in the X-register. |
| NO RECORD | The record pointer is positioned at record 0. Use IRCPT to set it to record 1. |
| NO WORKFL | There is no current or working ASCII file selected. Use RSTAF to select a working file. |

---

MRCPT Move ReCord PoinTer
XROM 06,31

The record pointer is moved by the number of records specified in the X-register. The sign of the X-register determines whether the record pointer is advanced, or moved backwards. If the X-register is negative, the record pointer is moved back the specified number of records, else it is advanced by the number of records specified in the X-register.
The character pointer is not changed, unless the record, that is indicated by the new record pointer, is shorter than the character pointer indicates. In this case the character pointer is set to the last character in the new record.

The X-register contains the number of records the record pointer has to be moved. If the X-register is negative the record pointer is decreased, else it is increased the specified number of records.

The record pointer is moved the specified number of records into the specified direction.

| | |
|---|---|
| DATA ERROR | The X-register contains "0" as input. This should be at least 1. Change it and try again. |
| NONEXISTENT | An illegal number, less than -999 or more than +999, was specified in the X-register. |
| NO RECORD | The record pointer is positioned at record 0. Use IRCPT to set it to record 1.<br>The newly selected record does not exist. |
| NO WORKFL | There is no current or working ASCII file selected. Use RSTAF to select a working file. |

---

NAMAF NAMe Ascii File
XROM 06,32

This function returns the name of the current working ASCII file to Alpha.



None.

Alpha contains the name of the working ASCII File.

| | |
|---|---|
| NO WORKFL | There is no current or working ASCII file selected. Use RSTAF to select a working file. |

---

NAMDF NAMe Data File
XROM 06,33

This function returns the name of the current working Data File to Alpha.

None.

Alpha contains the name of the working Data File.

NO WORKFL There is no current or working Data file selected. Use RSTDF to select a working file.

---

NFLD? Number of FieLDs ?
XROM 06,34

The number of fields in the record where the record pointer is positioned at is recalled to the X-register.

None.

The X-register contains the number of fields in the current record. If the stack lift was enabled, the stack will be raised and the value in the T-register lost. If the stack lift was disabled, the X-register will be overwritten by the number of fields in the current record.

NO RECORD The record pointer is positioned at record 0. Use IRCPT to set it to record 1.
NO WORKFL There is no current or working ASCII file selected. Use RSTAF to select a working file.

---

NREC? Number of RECords ?
XROM 06,35

The number of records in this file is recalled to the X-register.

None.

The X-register contains the number of records in this file. If the stack lift was enabled, the stack will be raised and the value in the T-register lost. If the stack lift was disabled, the X-register will be overwritten by the number of records in this file.



NO WORKFL There is no current or working ASCII file selected. Use RSTAF to select a working file.

---

NUMREC get NUMber in RECord
XROM 06,36

Starting at the character pointer the current record is searched for a number. search is stopped when a number is found and read back to the X-register, or the end of the record is reached.
Numbers must satisfy a certain format to be recognized as a number. This format and special characters and/or treatments of certain characters are discussed in detail.
A number always starts with a digit or a minus sign. During the search all other characters are ignored until one of these

characters are found. This implies that a fractional number ought to start with a zero digit or a minus sign.
When a digit or a minus sign is found the function tries to build a number of the next characters. If the number starts with a minus sign the next character should be a digit, otherwise the minus sign is ignored.
After the first regognized character only digits, decimal points, commas and the "E" are legal characters.
Digits are placed into the number as they are encountered. Only the first ten digits are used to build the mantissa. All surplus digits are ignored.
The mantissa digit string may contain periods and/or commas. These are treated according to the setting of user flag 28.
When flag 28 is cleared, the comma will act as the decimal point in the mantissa. In this case the period serves as a digit seperator mark.
If on the contrary flag 28 is set, the period will act as the decimal point and the comma as a digit seperator mark. For further remarks on this grouping see also your HP-41 owners manual.
Once the decimal point has been encountered, all following commas and periods are treated as terminators of the number string.
Only the "E" and digits are legal characters now. Digits will be added to the mantissa until ten digits have been used. All following digits are ignored. The "E" indicates the start of the exponent field of a number.
This exponent may start with a minus sign and only contains 1 or 2 digits. All surplus digits are ignored.
A number is terminated whenever a character is encountered that is none of the above described characters, or one of these characters in an illegal place.
When the end of the record is reached during search zero is returne to the X-register. To be able to distinct this from the value zero in a record user flag 17 will be cleared when the end of the record is reached, else it is set. Compare this with GREC etc.

None.

The X-register contains the value of the first number found in



the record. If no number is encounterd before the end of the record is reached, zero is returned. If the stack lift was enabled, the stack will be raised and the value in the T-register lost. If the stack lift was disabled, the X-register will be overwritten by the found number.
The character pointer is set to the character following the terminating character. If the end of the record is reached during the search, the record pointer is set to the next record and the character pointer is placed at the first character of this record.
User flag 17 is cleared when the end of the record is reached during search, else it is set

| | |
|---|---|
| NO RECORD | The record pointer is positioned at record 0. Use IRCPT to set it to record 1. |
| NO WORKFL | There is no current or working ASCII file selected. Use RSTAF to select a working file. |

---

OUTCHR OUTput CHaRacters
XROM 06,37

The number of characters specified in the X-register in the current record is sent to the selected device on the HP-IL loop. The transfer is started at the current character pointer position and ends when the specified number of characters has been sent, or the end of the record is reached.
User flag 17 determines the way the transferred characters are terminated. When user flag 17 is clear a CR/LF is sent after the last character. This causes a printer to return the printing head to the start of the next line. If user flag 17 is set, no special terminating sequence is sent.
The Y-register is used to facilitate formatted output to any device on the HP-IL loop. This register controls the total number of characters transferred to the selected device.
If the number of characters in the Y-register is less then the number of characters to be sent, nothing unusual will happen. The field is just sent as it is eventually terminated with a CR/LF.
When there are more characters specified in the Y-register then there are specified in the X-register these characters are padded with blanks to make a output string with total length equal to the number specified in the Y-register.
The sign of the Y-register determines whether the output string is followed by the extra blanks or preceded. If the number in the Y-register is negative, the transferred characters are followed by the extra blanks, otherwise it is preceded by the blanks.

The Y-register contains the width of the output column.
The sign specifies left or right adjusted output. When the number is negative the output will be left justified in the column and when it is positive the output is right justified.

The specified number of characters in the current record are sent to the selected device on the HP-IL loop in the format specified in the Y-register.



The character pointer is set to the character following the last sent character. If there are less characters left then specified in the X-register, the record pointer is set to the next record and the character pointer is placed at the first character of this record.

| | |
|---|---|
| DATA ERROR | The X-register contains "0" as input. This should be at least 1. Change it and try again. |
| NO HPIL | The HP82160 HP-IL Module is not connected to the HP-41. Connect the HP-IL Module, switch on the HP-41 again and re-execute the OUTCHR function. |

| | |
|---|---|
| NONEXISTENT | An illegal number, less than -999 or more than +999, was specified in the X-register or Y-register. |
| NO RECORD | The record pointer is positioned at record 0. Use IRCPT to set it to record 1. |
| NO WORKFL | There is no current or working ASCII file selected. Use RSTAF to select a working file. |

---

OUTFLD OUTput FieLD
XROM 07,38

The field specified in the X-register in thecurrent record is sent to the selected device on the HP-IL loop. The transfer is started at the first character of the specified field and ends when the end of the field is reached.
User flag 17 determines the way the transferred field is terminated. When user flag 17 is clear a CR/LF is sent after the last character. This causes a printer to return the printing head to the start of the next line. If user flag 17 is set, no special terminating sequence is sent.
The Y-register is used to facilitate formatted output to any device on the HP-IL loop. This register controls the total number of characters transferred to the selected device.
If the number of characters in the Y-register is less then the number of characters to be sent, nothing unusual will happen. The field is just sent as it is eventually terminated with a CR/LF.
When there are more characters specified in the Y-register then there are in the specified field the field will be padded with blanks to make a output string with total length equal to the number specified in the Y-register.
The sign of the Y-register determines whether the output string is followed by the extra balnks or preceded. If the number in the Y-register is negative, the field is followed by the extra blanks, otherwise it is preceded by the blanks.

The Y-register contains the width of the output column.
The sign specifies left or right adjusted output. When the number is negative the output will be left justified in the column and when it is positive the output is right justified.

The specified field in the current record is sent to the selected device on the HP-IL loop in the format specified in the Y-register.



The character pointer is advanced to the first character of the next field. If the specified field was the last field of the record, the record pointer is set to the next record and the character pointer is placed at the first character of this record.

| | |
|---|---|
| DATA ERROR | The X-register contains "0" as input. This should be at least 1. Change it and try again. |
| NO FIELD | There is no such field in the record. Specifie an |

existing field in the X-register.

| | |
|---|---|
| NO HPIL | The HP82160 HP-IL Module is not connected to the HP-41. Connect the HP-IL Module, switch on the HP-41 again and re-execute the OUTFLD function. |
| NONEXISTENT | An illegal number, less than -999 or more than +999, was specified in the X or Y-register. |
| NO RECORD | The record pointer is positioned at record 0. Use IRCPT to set it to record 1. |
| NO WORKFL | There is no current or working ASCII file selected. Use RSTAF to select a working file. |

---

OUTREC OUTput RECord
XROM 06,39

The current record is sent to the selected device on the HP-IL loop. The transfer is started at the character the character pointer is positioned at and ends when the end of the record is reached.
User flag 17 determines the way the transferred record is terminated. When user flag 17 is clear a CR/LF is sent after the last character. This causes a printer to return the printing head to the start of the next line. If user flag 17 is set, no special terminating sequence is sent.
The Y-register is used to facilitate formatted output to any device on the HP-IL loop. This register controls the total number of characters transferred to the selected device.
If the number of characters in the Y-register is less then the number of characters to be sent, nothing unusual will happen. The record is just sent as it is eventually terminated with a CR/LF.
When there are more characters specified in the Y-register then there are in the current record, the record will be padded with blanks to make a output string with total length equal to the number specified in the Y-register.
The sign of the Y-register determines whether the output string is followed by the extra balnks or preceded. If the number in the Y-register is negative, the record is followed by the extra blanks, otherwise it is preceded by the blanks.

The Y-register contains the width of the output column.
The sign specifies left or right adjusted output. When the number is negative the output will be left justified in the column and when it is positive the output is right justified.

The current record is sent to the selected device on the HP-IL loop in the format specified in the Y-register.



The record pointer is advanced to the next record and the character pointer is set to the first character in this record.

| | |
|---|---|
| NO HPIL | The HP82160 HP-IL Module is not connected to the HP-41. Connect the HP-IL Module, switch on the HP-41 again and re-execute the OUTREC function. |
| NO RECORD | The record pointer is positioned at record 0. Use |

IRCPT to set it to record 1.

NO WORKFL — There is no current or working ASCII file selected. Use RSTAF to select a working file.

---

RCHPT — Recall CHaracter PoinTer
XROM 06,40

The value of the character pointer is recalled to the X-register. The first character of a record is indicated with a character pointer value of one.

None.

The X-register contains the value of the character pointer. If the stack lift was enabled, the stack will be raised and the value in the T-register lost. If the stack lift was disabled, the X-register will be overwritten by the value of the character pointer.

NO RECORD — The record pointer is positioned at record 0. Use IRCPT to set it to record 1.

NO WORKFL — There is no current or working ASCII file selected. Use RSTAF to select a working file.

---

RCHR — Recall CHaraCters
XROM 06,41

From the current record the specified number of characters in the X-register is appended to Alpha. Transfer starts at the current character in the record. When during transfer the end of the record is reached user flag 17 will be cleared to indicate the end of the record else user flag 17 is set to indicate that the end of the record is not reached yet.
The number of characters transferred depends on the empty space in Alpha and the number of characters specified in the X-register. Transfer stops after Alpha is filled or the end of the record is reached.

The X-register specifies the number of characters to be appended to Alpha. Transfer starts at the current character position.

The specified number of characters is appended to the contents of Alpha. The character pointer is set to the last transferred character. If the last transferred character is the last character of the record, the record pointer is advanced to the next record.



When the last character of the record is reached during transfer user flag 17 is clear, else it will be set.

DATA ERROR — The X-register contains "0" as input. This should be at least 1. Change it and try again.

| | |
|---|---|
| NONEXISTENT | An illegal number, less than -999 or more than +999, was specified in the X-register. |
| NO RECORD | The record pointer is positioned at record 0. Use IRCPT to set it to record 1. |
| NO WORKFL | There is no current or working ASCII file selected. Use RSTAF to select a working file. |

---

RESAF RESize Ascii File
XROM 06,42

The named ASCII file is resized to the number of registers specified in the X-register. When the ASCII file is resized to a larger size, the following files in the datablock containing the file to be resized are moved upwards , to make place for the larger file. If on the contrary the ASCII file is sized to a smaller number of registers, the following files are moved downwards to maintain as much space as possible.

Warning
Be carefull when resizing the ASCII file to a smaller size. If the size is becoming less then is needed to hold all records, data will be lost. Data in the ASCII file is always lost in whole records at a time. In other words if the named ASCII file is resized such, that the last character of the last record would be lost, the whole record is lost.

The X-register should contain the new size of the ASCII file in registers. Note that this value is the new size, not the number of registers to remove.

The working ASCII and Data file becomes undefined.

| | |
|---|---|
| DATA ERROR | The X-register contains "0" as input. This should be at least 1. Change it and try again. |
| FL TYPE ERR | The file specified in Alpha is not an ASCII type file. Use the FLDIR function to check that the file exists. |
| NAME ERROR | An illegal file name was specified in Alpha. File names of 1 to 7 characters are allowed. |
| NO FL NAME | No file name was specified in Alpha. |
| NONEXISTENT | An illegal number, less than -999 or more than +999, was specified in the X-register. |
| NO ROOM | The specified length for the file is too long for the amount of free registers remaining in the RSU-Data Block. Either free additional data registers by deleting a file (using DELFL), or resize another file(s). If only one data block is available, the RLEFT? can be used to check the number of free regsters remaining. |



| | |
|---|---|
| NOT A FILE | The name specified in Alpha is not that of an existing ASCII file. |

---

RFLD Recall FieLD
XROM 06,43

From the current record a specified field is appended to Alpha. Normally the first character used is the first character of the specified field. If however user flag 17 is set, transfer starts at the current character in the record. This facilitates transfer of fields longer then 24 characters. When the end of the field is reached user flag 17 will be cleared to indicate the end of the field, else user flag 17 is set to indicate that only part of the field is transferred.
The number of characters transferred depends on the empty space in Alpha and the length of the field. Transfer stops after Alpha is filled, the end of the field or the end of the record is reached.

The X-register specifies the field to be appended to Alpha. User flag 17 determines whether transfer is started at the first character of the field, or at the current character position. When clear, transfer starts at the fist character of the field.

The contents of the field is appended to the contents of Alpha. The character pointer is set to the last transferred character. If the last transferred character is the last character of the record, the record pointer is advanced to the next record.
When the last character of the field is reached during transfer user flag 17 is clear, else it will be set.

| | |
|---|---|
| DATA ERROR | The X-register contains "0" as input. This should be at least 1. Change it and try again. |
| NO FIELD | There is no such field in the record. Specifie an existing field in the X-register. |
| NONEXISTENT | An illegal number, less than -999 or more than +999, was specified in the X-register. |
| NO RECORD | The record pointer is positioned at record 0. Use IRCPT to set it to record 1. |
| NO WORKFL | There is no current or working ASCII file selected. Use RSTAF to select a working file. |

---

RLEN? Record LENgth ?
XROM 06,44

The length of the record where the record pointer is positioned at is recalled to the X-register. This is the total length of the record including the delimiter bytes.

None.

The X-register contains the length of the current record. If the stack lift was enabled, the stack will be raised and the value in



the T-register lost. If the stack lift was disabled, the X-

register will be overwritten by the length of the current record.

| | |
|---|---|
| NO RECORD | The record pointer is positioned at record 0. Use IRCPT to set it to record 1. |
| NO WORKFL | There is no current or working ASCII file selected. Use RSTAF to select a working file. |

---

RRCPT Recall ReCord PoinTer
XROM 06,45

Recalls the value of the record pointer in the working ASCII file to the X-register.
The record pointer and character pointers are not changed.

Warning
If a new ASCII file is specified the record pointer is placed at record 0. This is not an useable record. Record 0 contains information specific for the working file. However, RRCPT correctly gives 0 for the value of the current record.

None.

The X-register contains the number of the current record. If the stack lift was enabled, the stack will be raised and the value in the T-register lost. If the stack lift was disabled, the X-register will be overwritten by the number of the current record.

| | |
|---|---|
| NO WORKFL | There is no current or working ASCII file selected. Use RSTAF to select a working file. |

---

RREC Recall RECord
XROM 06,46

The contents of the record in which the record pointer is positioned is appended to Alpha from the position of the character pointer to either the last character in the current record or so many characters from the character pointer untill Alpha is completely filled (24 characters).
When during transfer the end of the record is reached user flag 17 will be cleared to indicate the end of the record else user flag 17 is set to indicate that the end of the record is not reached yet.

None.

Part of the record is appended to Alpha. The character pointer is set to the last transferred character. If the last transferred character is the last character of the record, the record pointer is advanced to the next record.
When the last character of the record is reached during transfer user flag 17 is clear, else it will be set.

| | |
|---|---|
| NO RECORD | The record pointer is positioned at record 0. Use IRCPT to set it to record 1. |
| NO WORKFL | There is no current or working ASCII file selected. Use RSTAF to select a working file. |

---

RSTAF ReSet Ascii File
XROM 06,47

The file specified in Alpha is made the working ASCII file. The record pointer is set to record 0 and the character pointer is not defined.

Warning
Record 0 contains information about the working file. Therefore this record is not useable and another record must be selected before any file handling can take place.

Alpha contains the name of the new working file.

| | |
|---|---|
| FL TYPE ERR | The file specified in Alpha is not an ASCII type file. Use the FLDIR function to check that the file exists. |
| NAME ERROR | An illegal file name was specified in Alpha. File names of 1 to 7 characters are allowed. |
| NO FL NAME | No file name was specified in Alpha. |
| NOT A FILE | The name specified in Alpha is not that of an existing ASCII file. |

---

SCHPT Set CHaracter PoinTer
XROM 06,48

The character pointer is set to the specified character in the current record.
The first character in a record is character number 1 and the maximum value of the character pointer is 255.

The X-register contains the new value of the character pointer.

None.

| | |
|---|---|
| DATA ERROR | The X-register contains "0" as input. This should be at least 1. Change it and try again. |
| END OF REC | The character pointer would be positioned over the end of the record. |
| NONEXISTENT | An illegal number, less than -999 or more than +999, was specified in the X-register. |
| NO RECORD | The record pointer is positioned at record 0. Use IRCPT to set it to record 1. |
| NO WORKFL | There is no current or working ASCII file selected. Use RSTAF to select a working file. |

---

SFPT Set Field PoinTer

XROM 06,49

The character pointer is set to the first character of the specified field in the working record. The first field in a record is numbered 1.

The X-register specifies the field to set the character pointer to the first character.

None.

| | |
|---|---|
| DATA ERROR | The X-register contains "0" as input. This should be at least 1. Change it and try again. |
| NO FIELD | There is no such field in the record. Specifie an existing field in the X-register. |
| NONEXISTENT | An illegal number, less than -999 or more than +999, was specified in the X-register. |
| NO RECORD | The record pointer is positioned at record 0. Use IRCPT to set it to record 1. |
| NO WORKFL | There is no current or working ASCII file selected. Use RSTAF to select a working file. |

---

SRCPT Set ReCord PoinTer
XROM 06,50

The record pointer is set to the record specified in the X-register.
The character pointer is not changed, unless the record, that is indicated by the new record pointer, is shorter than the character pointer indicates. In this case the character pointer is set to the last character in the new record.

The X-register contains the new record pointer.

None.

| | |
|---|---|
| DATA ERROR | The X-register contains "0" as input. This should be at least 1. Change it and try again. |
| NONEXISTENT | An illegal number, less than -999 or more than +999, was specified in the X-register. |
| NO RECORD | The record pointer is positioned at record 0. Use IRCPT to set it to record 1.<br>The newly selected record does not exist. |
| NO WORKFL | There is no current or working ASCII file selected. Use RSTAF to select a working file. |

---

ST<>DF STack <> DataFile
XROM 06,51

The stack registers T-Z-Y-X an LastX are exchanged withe the registers in the datafile starting at the current position of the pointer.
Opposite to the function EXST the pointer is not changed.

None.

The stack and five registers in the RSU file are exchanged.

| | |
|---|---|
| END OF FL | The data file pointer is positioned less that five registers from the end of the selected file. Reset the pointer to a value less than the size of the data file, or increase the file size with the RSZDF function. |
| NO WORKFL | There is no current or working ASCII file selected. Use RSTAF to select a working file. |

---

SWPREC SWaP RECord
XROM 06,52

The current record and the record specified in the X-register are swapped. The record pointer is not moved. The character pointer is set to the first vharacter of the record.

Warning.
To be able to swap two records with different lengths as quick as possible the shortest record is lengthened to the same length as the longest record. It might be possible that there is not enough room in the file to do this. In this case the file should be made longer, or records should be deleted.

In the X-register the record to swap the current with is specified.

The character pointer is set to the first character in the current record.

| | |
|---|---|
| DATA ERROR | The X-register contains "0" as input. This should be at least 1. Change it and try again. |
| FILE FULL | There is not enough free space in the file to extend the shortest record to the same length as the longest. Lengthen the file with RESAF or delete records. |
| NONEXISTENT | An illegal number, less than -999 or more than +999, was specified in the X-register. |
| NO RECORD | The record pointer is positioned at record 0. Use IRCPT to set it to record 1.<br>The specified record does not exist. |
| NO WORKFL | There is no current or working ASCII file selected. Use RSTAF to select a working file. |

---

SZ?AF SiZe ? AsciiFile
XROM 06,53

The length of the working ASCII file in registers is returned to the X-register.

None.

The X-register contains the length of the working file. If the stack lift was enabled, the stack will be raised and the value in the T-register lost. If the stack lift was disabled, the X-register will be overwritten by the length of the working file.

| | |
|---|---|
| NO WORKFL | There is no current or working ASCII file selected. Use RSTAF to select a working file. |

---

X<>CHR X <> CHaraCter
XROM 06,54

This function exchanges the character pointed to by the character pointer with a character specified in the X-register. The character in the X-register is given as its decimal ASCII value. The character returned from the current record is in the same decimal format.

Warning.
Be carefull with character "_", decimal 95. This character is used by the text editor as a cursor. When this character is encountered unpredictable results are possible. Never use this character in a text file when the text editor will be used with this text file.

The X-register holds the decimal representation of the character to be entered in the working file.

The X-register holds the decimal representation of the character pointed to by the character pointer.
The pointers are not changed.

| | |
|---|---|
| DATA ERROR | The X-register contains "0" as input. This should be at least 1. Change it and try again. |
| NONEXISTENT | An illegal number, less than -999 or more than +999, was specified in the X-register. |
| NO RECORD | The record pointer is positioned at record 0. Use IRCPT to set it to record 1. |
| NO WORKFL | There is no current or working ASCII file selected. Use RSTAF to select a working file. |

---

?EOF ? End Of ascii File
XROM 06,55

This function tests if the record and character pointer are positioned to the end of the file. If this is the case the result of the function is true, else it is false.

None.

When manual executed YES if at the end of file, NO if not.
Under program control it skips the next step if false, else it